DOCUMENT RESUME

ED 270 621 CE 044 576

AUTHOR Leonard, Janet
TITLE Computers in Language and Literacy Work.
INSTITUTION Adult Literacy and Basic Skills Unit, London (England).; Inner London Education Authority (England).
REPORT NO ISBN-0-906509-599
PUB DATE Sep 85
NOTE 184p.
AVAILABLE FROM Adult Literacy & Basic Skills Unit, Kingsbourne House, 229/231 High Holborn, London, England WC1V 7DA (2.50 pounds).
PUB TYPE Information Analyses (070) -- Guides - Non-Classroom Use (055)

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Adult Basic Education; *Adult Literacy; *Computer Assisted Instruction; *Courseware; Educational Media; Educational Trends; *English (Second Language); Foreign Countries; Futures (of Society); Instructional Materials; Language Acquisition; *Literacy Education; Material Development; Second Language Learning; Trend Analysis
IDENTIFIERS *Computer Assisted Language Learning; United Kingdom

ABSTRACT

This book surveys currently available software for English language learners and identifies future directions in the development of such software that are likely to prove fruitful. Discussed first are ways in which computers add to the learning experience. Stand-alone programs are compared with authoring systems, and "purpose-made" and "borrowed" software and software for students and teachers are contrasted. Discussed next are the following types of software currently available for English language learners: alphabet-based, letter-word manipulation, word-based, text-based, database and rule, structure-based, adventure game, simulation and role play, creative writing aid, and "total-package" programs. The next chapters deal with increasing the range of programs in the Computer-Assisted Language Learning (CALL) programs: making them more user friendly; selecting appropriate currently available and future hardware for use with CALL programs; involving English-as-a-foreign-language, English-as-a-second-language, and literacy teachers in CALL program development; and making the most of CALL opportunities. Appended to the book are a list of acronyms, bibliography, directory of CALL programs, directory of software publishers, and list of useful addresses. (MN)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

COMPUTERS IN LANGUAGE AND LITERACY WORK

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

A Wells

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED270621

CE044576

ALBSU
Adult Literacy & Basic Skills Unit

ilea

© ALBSU, ILEA and M.J. Leonard

All rights reserved. No part of this publication may be photocopied recorded or otherwise reproduced, stored in a retrieval system or transmitted in any form or by any electronic or mechanical means without the prior permission of the copyright owners.

Published September 1985

ISBN 0 906509 599

Design: Studio 21

Published November 1985
Printed by Whitstable Litho

Contents

Preface

Introduction

What can computers add to the learning experience? 9
Some preliminary questions and answers 10
What kind of language-learning activity is the computer useful for? 13

Chapter 1

Review of current software for English language learners 16
"Stand-alone" programs v. "Authoring" systems 16
Purpose-made software v. "borrowed" software 17
"Computers for students" v. "Computers for teachers" 18
Program types 18
1. Alphabet-based programs 19
2. Letter/word manipulation programs 20
3. Word-based programs 23
4. Text-based programs 28
5. Database and rule programs 34
6. Structure-based programs 39
7. Adventure games 40
8. Simulation and role-play 42
9. Aids to creative writing 47
10. "Total package" software 49

Chapter 2

Increasing the range of programs in CALL 51
Simulated dialogue 52
Simulation 54
Adventure games 62
Word-processing 66
Programs basd on word-processing techniques 70
Audio linking 75
Special education — contributions to CALL? 78
Databases 78
Communicating with the outside world 80
Which way for the future? 81

Chapter 3

Making CALL programs "user-friendly" 83
Documentation and program instructions 85
Menu-driven discs and programs 87
Program security 88
Checking input 89
Responding to incorrect inputs 92
Colour, graphics and sound 94
Printers 97
Record-keeping 98
Software under student-control 100
Accessibility of software 102
Drawing conclusions 104

Chapter 4

Hardware — today and tomorrow 105
Choosing computer hardware 106
Choosing hardware: financial considerations 108
Choosing hardware: initial decisions 110
Future developments in CALL hardware 115
Computers and the spoken word 119
Computers and video 122
Programming with the chips of the future 125

Chapter 5

Teacher-involvement and program-development 128
EFL, ESL and Literacy — different requirements? 129
CALL in EFL and ESL 129
Computers in Adult Literacy 134
Student involvement 137
What teachers want 138
CALL programming — a cottage industry? 139
A CALL watchdog 140
Who should develop CALL programs? 141
Is CALL a worthwhile investment? 143

Conclusion

Making the most of CALL opportunities 145

Appendix A
List of acronyms 148

Appendix B
Bibliography 149

Appendix C
Directory of CALL programs 153

Appendix D
Directory of software publishers 178

Appendix E
Useful addresses 184

Preface

"Computers in Language and Literacy Work" is intended to stimulate computer assisted learning related to language and reading and writing skills with adults and with young adults in further education. One of the major difficulties faced by enthusiastic teachers and tutors who want to become involved in computer assisted learning is knowing both what is available and where to most effectively direct their energies.

This book seeks to help by:

☐ surveying language learning related software

☐ identifying future directions which are likely to prove fruitful

It should provide a comprehensive guide to the field as well as a straightforward book for the non expert.

"Computers in Language and Literacy Work" was written by Janet Leonard as part of a two term secondment granted to her by the ILEA from her job at Westminster College. Support and help is acknowledged from many staff in colleges and adult education institutes, from the Inner London Educational Computing Centre and from the ILEA Inspectorate.

Introduction

What can computers add to the language-learning experience?

The computer is just another learning resource. It adds a new dimension to language-learning, but it does not replace anything — least of all the teacher!

And yet computer programs, carefully tailored to suit particular groups of students, will soon be able to offer facilities equal to, and in many cases surpassing, any one of the other resources teachers now use with their students. Seymour Papert, in his book *Mindstorms* (Papert, 1982), writes:

> The computer is the Proteus of machines. Its essence is its universality, its power to stimulate. Because it can take on a thousand forms and can serve a thousand functions, it can appeal to a thousand tastes.

There are already language-based programs available for some machines, but their authors would be the first to admit that what has been done so far has merely scratched the surface of what *can* be achieved within a very few years, given technology which is improving rapidly, and costs which are falling equally rapidly.

This study will outline:

☐ what software is available to teachers now;

☐ what are the likely software developments in the near future;

☐ what kind of standards we should expect from CALL (Computer-Assisted Language-Learning) programs;

☐ what limitations are imposed by currently-available choices of hardware, with some suggestions as to how the problems may be minimised;

☐ what the future might hold in terms of new technology and new applications.

The study will concern itself principally with CALL for adult learners of English as a Foreign Language (EFL) and English as a Second Language (ESL). However, a great deal of the subject-matter will be of interest in other subject-areas.

Much CALL software can be applied in Adult Literacy teaching, for example, and conversely programs developed specifically for Literacy work could have direct applications in ESL or EFL environments.

Where teachers can supply their own data to a basic program-frame (that is, via "authoring systems"), programs can be tailored to suit the needs of a particular group of students (or even an individual student), and this can open up great opportunities for the exchange of software across what are normally impassable barriers — ESL software can be applied in mainstream English, in primary as well as secondary schools, and there could also be traffic in the opposite direction.

Furthermore, CALL can draw on the whole range of CAL (Computer-Assisted Learning) software across the curriculum; there is no reason why an adult student, whose main object is language-improvement, should not acquire English language via other interesting subjects — history, chemistry, music, literature or whatever.

Some preliminary questions and answers

"Students can do that kind of thing with a pencil and paper. Why use a computer?"

A computer is just another weapon in the armoury of teaching materials. Some students learn better by chalk-and-talk; some enjoy audio-cassettes while others can't bear to wear headphones; some have difficulties with writing and would welcome anything that helps them to bypass the pen in their efforts to put words on paper; some learn well from games, others hate them. It is a case of "horses for courses". Evidence so far suggests that even mundane drill-and-practice goes down better when presented on a computer than via a textbook, while the more imaginative applications of the computer have no paper-and-pencil equivalents.

Of course, there were probably teachers in the past who asked, "Why print a book when you can get hold of illuminated manuscripts so easily?"

"A computer de-humanises the learning process."

The advent of the language-laboratory was regarded as a turning-point in language-learning practice. Nowadays, we are less likely to be impressed by row upon row of students, each in a little box, isolated from everyone else by headphones, being talked at by a native speaker who can often be satisfied only by conversations not far removed from "the pen of my aunt is on the table of my uncle". Now, there's a de-humanising process for you!

Contrast this with the use of a computer: the computers are in a room which contains other learning materials, so students make a choice as to which kind of study they wish to pursue. There are far fewer computers than students, so each machine is being used by a pair of students working together, or by groups of three to five. The programs require intelligent thought on the part of the user before a response is made, so each little group of participants is exchanging opinions (in English) on what the next response should be.

These scenarios are obviously a "worst-case" and a "best-case". Not all language-laboratory work is dull, and not all computer programs are stimulating. But it does demonstrate that it isn't so much the equipment that matters, as the way that you use it. If we insist on treading the same path with computers as we did with audio materials, we could end up with one-to-one computer-laboratories, each student isolated from the others, while we all work through Unit 5 of our coursebook microcomputer-drills. But it needn't be like that!

"Students will waste time with the machinery instead of learning English, and they won't know the keyboard - and I'm no good with machines anyway."

Well-designed software should require very little technical knowledge on the part of the user. For students (or teachers) whose aim is to benefit from ready-made programs, five minutes' instruction on loading up the machine should be more than adequate. The first thing to learn about computers is that they are very difficult to break, and it doesn't really matter if you press the wrong key, so a bit of experimentation does no harm. If several students are learning at the same time, some useful interchanges might be generated on the theme of "giving instructions" and "offering advice"!

Knowledge of the keyboard is handy, but not essential. Many students will already know the keyboard, and those who do not will quickly pick up enough to get by.

As for the teacher's own technical knowledge, suffice it to say that computers "go wrong" far less often than, say, language-laboratories, and there will usually be at least one student in the group who can help with any hitches that arise.

"It takes hundreds of hours to develop a computer program that lasts a few minutes - that's not cost-effective."

It takes hundreds of hours to prepare and record a set of tapes, but thousands of students use them, often for many years, which *is* cost-effective.

Computer programs can be mass-produced and used all over the world, so they can be as economically viable as any other teaching matter. But the types of CALL program which involve text and/or vocabulary often have the additional advantage that they can be readily modified (via an "authoring system" which allows teachers to add a personal choice of data to the program) to suit changing circumstances. How often do teachers stop using a particular book, or text from a book, because the material, though good in concept, has become out of date? How often does a teacher regret the presence of one unsatisfactory example in an otherwise good exercise? Much of the data-oriented computer software for CALL can be altered relatively quickly, unsatisfactory data removed and new data inserted, by a teacher with no programming experience. And this can be done not once, but as often as is necessary to keep the program up to date. Moreover, if CALL publishers follow the example of publishers of business software, major packages would be sold with an automatic right to future revisions of the program itself at a minimal cost.

"I might ultimately lose my job."

People said that art was dead when the camera was invented. Yet artists still thrive in their own right, and even make use of and benefit from photography.

People said that language-laboratories would mean an end to language-teaching as we know it, when in fact the result was to create *more* jobs for teachers, scriptwriters, actors, technicians and so on.

There is no reason why computers should put teachers out of work, any more than the language-laboratory did. To quote John Higgins, a pioneer CALL programmer: "If you think you can be replaced by a computer, you probably should be."

"Computers will take over."

Some teachers have qualms about the advent of the computer. Their fears, like the fears about the language-laboratory twenty years ago, will prove groundless: the computer is just another teaching-aid. Some teachers will take to it, others will not. Some students will love it, others will avoid it like the plague. But one of the chief advantages of the computer over the language-lab will be that the computer can be integrated with other activities, so that students can *choose* it, rather than having it forced upon them because the whole class is "doing a lab lesson".

Also, as programs become more sophisticated, and teachers become more adept at applying them imaginatively, the computer will come to be used as a means to an end, rather than as an end in itself.

Newcomers to microcomputing should resist the temptation to think of the computer as a teaching-machine, just another experiment in programmed learning. Few programs attempt to usurp the teacher's role. On the other hand, even fewer lend themselves to whole-class activities in which the teacher functions as a demonstrator or key-presser, while the students are passive, and captive, observers.

A far more productive use of the computer, with a wide-ranging and carefully-chosen selection of programs, is to provide a valuable source of self-access study materials for students who choose to use them. Some of those materials will demand a one-to-one relationship — a single student working at the machine — but the vast majority will either require, or at least be improved by, the participation and co-operation of a small group working together. Some of the programs will have been specifically written with language skills in mind, but many others may be drawn from sources as far removed from language learning as, say, home-computer games or O-Level Physics. Whatever the source or overt aim of the program, the communication which takes place among the users around the computer is always a vital by-product, and often the most important reason for using the program with language students.

Ideally, the machine(s) will be sited in a "workshop" environment, where the computer will be one of a variety of activities available to students, and where a teacher will be on hand to help out if necessary. Alternatively, the micro may be a permanent part of the classroom furniture, to be used when required, possibly in group-work sessions. There may also be back-up machines in the library or other personal study areas.

The least acceptable siting of microcomputers for language work is in a computer-laboratory. The primary objection to computer-laboratories from the language learner's point of view is that the machines are divorced from other kinds of language activity; it requires a special effort and/or special timetabling to use the laboratory, and will probably mean that all the students are forced to participate in computer-based learning, whether they like it or not. Housing the computers in a separate room may also prevent them from being integrated into other kinds of language-work, and will almost certainly prevent their casual use to illustrate a point which arises during class work, or for remedial purposes.

The type of work which is most suited to the micro, and the type of environment in which computer work is most likely to be successful, both suggest that computers are not about to "take over". On the other hand, they can be invaluable in many ways as a "classroom assistant". Derrick Daines, in his contribution to *Exploring English with Microcomputers* (Chandler [Ed.], 1983), puts the computer firmly in its place:

> The turning point in my thinking came when I realised that it was a gross error to have the computer "centre-stage". The computer is a tool and like any tool its proper place is in the wings.

What kind of language-learning activity is the computer useful for?

Many people — even computer programmers — will tell you that computers are "good at" drill-and-practice, mechanical question-and-answer in a multiple-choice format, and anything else to which rigid rules can be applied.

Closer investigation of the phrase "good at", however, reveals that its real meaning in this context is that computers can easily be programmed to do these mechanical tasks, and that large numbers of multiple-choice tests in various guises can be churned out quickly and cheaply. It does *not* mean that the computer is only good for this type of activity — simply that anything more imaginative takes more time and care, and will therefore be longer in coming on to the market, and probably more expensive when it arrives.

The result is that much of what is available in the way of software today is unimaginative and mechanistic. And as Daniel Chandler observes (Chandler, 1984):

> Apart from the motivation of commercial interests to produce passive consumers rather than thinkers, probably the main reason for the scarcity of imaginative software and applications has been popular ideas about what computers are like. These ideas have already had a profound effect on our attitude to education.

Depressing as this analysis may seem, the computer — even in its present state of development, and with its present software — is in fact so versatile that it can be applied in a vast number of ways, limited only by a teacher's imagination. As micros become cheaper and bigger (bigger in terms of memory, that is, though smaller in physical size), applications which could not have been dreamt of even five years ago become possible. In terms of reading and writing activities, the micro can handle most things involving text manipulation, and can give freedom of response to the user, varying from a simple A/B/C/D choice, via the ability to cope with answers indicating varying degrees of probability, to the complete freedom of the word-processor.

But microcomputers have other talents in unexpected areas. One of their chief values is their facility for stimulating oral skills — not oral interchange between student and computer, but between student and student. A perennial problem for teachers, particularly where classes are large, is the difficulty of promoting discussion within groups of students. Even when the topic is interesting, motivation soon fizzles out, groups are left wondering what to say next, and the teacher ends up dashing from group to group bearing new ideas, after the manner of those entertainers who keep a dozen plates spinning simultaneously at the top of tall poles. The computer can do the simple job of ideas-promoter very well, providing a focus for discussion and throwing up new challenges as and when the group needs them.

Another feature of micros which has hardly been explored at all in terms of language-work, is the ease of connecting the computer to other equipment. An ordinary audio-cassette might be used in conjunction with a computer program to improve listening skills; the micro could be linked to a slide-projector to add a visual element (for basic vocabulary work, perhaps); different forms of input could be used to help students who are not familiar with a typewriter keyboard; interactive video will before long fall within the financial scope of many of us; and already there are the possibilities opened up by telephone links to external databases, such as Micronet and Prestel, to gain access to vast quantities of information on a wealth of subjects, not to mention using a "personal bulletin board" to produce students' own daily or weekly "newspapers" of events past and future, supplemented by film or book reviews or any topic of current interest.

Our young people are growing up in a society where computerisation is making itself felt more strongly than ever before, both in work and leisure activities. Many of our students, especially in the field of ESL, will want to learn about microcomputers for their own sake, and will be interested in acquiring basic computer skills, and more and more students coming into EFL and ESL will already have had some contact with micros through school. The Further Education Unit's handbook *Computer Literacy* (Lloyd, Taylor & West, 1984, 1) states:

> It should be stressed that the operation of a microcomputer in vocational preparation schemes should be closely tied to working situations. Motivation of trainees is vastly increased when the operation skills are learned using software which is closely related to their own career interest.

Many EFL students learn English solely because it enhances their employment prospects in their own countries. ESL students will often be taking computer-awareness or secretarial-skills courses alongside their language work, or will want to graduate to such courses as their competence in English improves. Language-teachers could capitalise on this growing motivation amongst students, by incorporating an element of "computer awareness" into their choice of programs, and offering a keyboard-tutor plus a selection of business applications programs (such as databases, financial spreadsheet and word-processing packages), alongside more traditional "English-language-learning" options. The programs chosen for this purpose could be standard packages for advanced students; but introductory packages, particularly for word-processing and database applications, would be useful as "tasters", offering most of the major features of the standard packages, but in a form digestible by those whose English is more limited.

These, in general terms, are some of the ways in which micros might be used in the teaching of English. The remainder of this study will be devoted to looking into these ideas in more detail, along with some of the hardware headaches which bedevil every application of microcomputing at the present time.

Chapter 1

Review of current software for English language learners

Large quantities of computer software are flooding both the educational and home-computer markets as publishers rush to take advantage of the computer-boom. The quality of these programs is often less impressive than the quantity, and it is tedious, time-consuming and expensive to sort the wheat from the chaff.

What is more, programs written specifically for language learners are few and far between, so in order to build a reasonable stock of software for language-learning applications, teachers must be prepared to extend the search. This means taking advantage of games and other home-computer applications, as well as programs written for other subject-areas in the educational world. Exploring these sources opens up a vast range of potentially interesting and useful software.

But those with responsibility for acquiring computer software for language-learning applications need to be able to choose wisely, and know how to avoid making expensive mistakes — whether they arise from the unsuitability of the material, or sheer bad programming, and whether the software is specifically geared to language students or not.

This chapter looks at the types of program which are now available, with particular reference to their suitability for language-learning applications, and attention is drawn to some of the features, desirable and undesirable, to look for in computer software. Therefore, although it is not by any means a comprehensive survey of individual programs, the remainder of the chapter should provide a potential purchaser with a good idea of what can be expected of the current generation of software, and the kind of questions to ask before making a purchase.

"Stand-alone" programs v "Authoring" systems

It is important to distinguish at the outset between these two categories of educational software, each of which has a place in language-learning, though it will be obvious that the second type is a great deal more flexible than the first.

The first category consists of "stand-alone" programs, where "what you buy is what you get", and there is no simple way (often, not even a difficult way) of adding to or altering the program data. These programs may be suitable, provided that the data supplied happens to be what you need. Let the buyer beware, however, because if the program comes with ten texts or ten sets of vocabulary, ready-made, it may be that only one of them will be useful to your students. This often happens with books, of course, but at least you are able to skim through a book before purchase to see whether an acceptable percentage is usable; generally speaking, the only contact you will have with a suite of programs before you buy them is the publisher's blurb!

The second software category is more useful, and consists of a program-skeleton into which teachers (or, in some cases, students) slot their own choice of text or vocabulary. Although what you buy is a suite of two programs (a "skeleton", which is the part the student sees, and an "authoring system", which helps you to enter your own data in the correct format), in effect, you are buying access to programs limited in number only by your own time and imagination.

Purpose-made software v. "Borrowed" software

Very little software has so far been written with ESL or EFL in mind, and even less for adult literacy work. And when you find a suitable program idea, you will probably discover that it runs on a make of machine which you have not got.

One important option open to teachers is to buy programs intended for other purposes, educational or otherwise. These may be, as already indicated, programs "borrowed" from the fields of computer-studies or office-skills, or they may be programs originally intended for other areas of mainstream education — English programs for secondary schools, or even science subjects, art or music. WHICH SALT?, for example, is a program about chemical analysis which involves interpreting the results of chemical tests, while EYE shows the user how the eye works. Although intended for secondary-school pupils, such programs may well be of interest to adult language-students, who would need to understand and manipulate the language in order to work through the experiments.

The problem with such programs is estimating their suitability for language-learning purposes. Programs can have all kinds of hidden pitfalls, and most commercial programs can not be "doctored" after purchase to make them more suitable. A prospective purchaser needs to be aware not only of the level of language used in the body of the program, but also the level and complexity of the instructions, the quantity and nature of the language required from the student (how much do students need to type in at the keyboard, and what level of language does the program demand of its users?), how tolerant the program is of "wrong" input, and whether the program is linked to course-material which may not be available to the student.

Bear in mind, also, that much of the software which seems most suitable on paper might be primary-school material, and may therefore turn out to be too easy, rather than too difficult. There may be graphic displays and sound-effects which are fine for children but offputting for adults; the data may be child-oriented and unalterable; even if the program itself is acceptable, there may be space-invaders or nursery-rhymes as "rewards" for correct answers.

A visit to a local school or college to see programs in use could prove fruitful, rather than throwing away money on a program which turns out to be less suitable in practice that it sounded on paper.

"Computers for students" v. "Computers for teachers"

A micro need not necessarily be manipulated by students in order to prove its worth. It may also be usefully employed by the teacher as an aid to the production of teaching materials, and this is another area of software which must be considered.

Apart from word-processing, which is a strictly do-it-yourself teaching-aid, a few other programs are available which help teachers to prepare class-sets of materials. These are mainly puzzle-programs, nearly all of which are of the "authoring" type, and they include anagram or "word-shuffling" puzzles, crosswords and wordsquares.

Program types

The old days of "programmed learning" have gone. Most computer programs do not attempt to "teach" anything; instead, the student follows the Chinese practice of "learning by doing", manipulating language in various ways from the level of the basic alphabet up to full-scale texts or stories.

The following are the chief groupings of program-types (from a language-learning point of view), although there are sometimes no perceptible boundaries between them:

1. Programs based on the alphabet, or on alphabetic ordering;
2. Exercises and games which manipulate letters or groups of letters, often for spelling-improvement;
3. Word-based programs — a wide variety, including everything from multiple-choice to contextualised word-games;
4. Text-based programs, including simple reading practice, reading for meaning, cloze and a host of others;
5. Programs which include a miniature database, or rules for manipulating language, and which allow the student to explore language by questioning the machine;
6. Structure-based programs — very few of these in the traditional style, but including programs where the structure is all-important, for example, telling the time;
7. Adventures;
8. Simulation and role-play;
9. Aids to creative writing, of which the chief one is the word-processor;
10. "Total package" software, where a suite of programs is linked with course-material.

An eleventh category, which is not included in the list because potentially it may be any or all of the other ten, is the "authoring language". Authoring languages, such as PILOT and MICROTEXT, are for teachers who do not want to get involved in the nitty-gritty of computer programming, but who need more flexibility than can be provided by authoring programs. Whereas an authoring *program* provides only the means for putting the "meat" of vocabulary or text on an existing "skeleton" program, an authoring *language* allows the teacher to create the bones of the program as well as the flesh. It is considerably more difficult to use an authoring language than an authoring program, but the results can be worthwhile.

1. Alphabet-based

Not many programs, other than those for very young children, deal with learning letter-shapes, and programs for very young children are generally unsuitable for older students. A suite of programs written for children with reading difficulties, however, provides a version which could well be used by adults. ABSHORT was designed to help a severely-handicapped spastic girl to learn the alphabet. It is of an "A is for Apple" type, but the objects used are not child-oriented. There are, naturally, no instructions in the program. A picture is drawn on the screen along with the corresponding letter of the alphabet, and if the user can press the matching key on the keyboard, the reward is a smiling face. Otherwise the face frowns. The letters on screen are, like those on the keyboard, all upper case, and the letters are presented in alphabetical order.

Another program from the same collection, unfinished at the time of writing, will (in the words of the distributors) "draw round script characters on the screen, slowly, and with the character growing in the same direction as the nib moves in writing. So far the alphabet has reached the letter "m" but the team is busy getting it completed." Other programs of this type which are already on the market include NOWRITE and LETTER FORMATION.

Manipulation of the alphabet, as opposed to simple letter-formation, comes in various forms. There are several versions of dictionary-type progams, which aim to strengthen the concept of alphabetical ordering: such a program contains a word-list, from which it draws a "secret" word at random, and the student is then invited to guess which word the computer has selected. When a guess is typed in, the computer compares the guess with the secret word, and reports back to the player whether its own word is earlier or later in the alphabet. The player then guesses again, and with each guess should home in on the secret word. A typical run of such a program would involve about a dozen tries before guessing the word correctly, and in the final stages might involve, for example, trying to find a word which comes after CARE and before CART in the alphabet. There should be an escape-route (a "cheat-routine") built into the program, for when the going gets tough!

2. Letter/Word manipulation

Based on word-meanings

Another dictionary-game, providing a transitional stage between the ordering-game and a real dictionary, involves using a mini-dictionary, either specially composed to suit a particular set of data, or a small standard dictionary. The on-screen display shows the initial letter of the word, along with a paraphrase of its dictionary definition. The user then hunts through the dictionary to find a word with that meaning, and types the word in at the keyboard. If the word is not correct, an additional clue may be offered, and the player sent back to the dictionary for another search. Apart from its general use with unstructured data, this kind of program, with a judicious choice of vocabulary, might be useful for learning to recognise, say, the noun, adjective, verb and adverb forms of the same root, and exploring differences in meaning and application between them.

A number of other games involve selecting or building words according to definitions. There are straight anagrams, where a clue is offered and the player unscrambles the word. There are programs involving "finding the odd-man-out" or "finding the opposite". There are "pair" games, involving matching words from two columns of vocabulary to make acceptable sets ("hand + bag = handbag" and " door + mat = doormat") or building acceptable words from the components offered (aus-tere, sensa-tion, re-vive). These ideas are very simple to program, and it is common to find seven or eight such programs offered as a "package deal" on one disc or tape. They should offer authoring facilities so that vocabulary can be altered according to need.

Based on spelling

Probably the most popular vocabulary game — for the producer, if not for the consumer! — is HANGMAN. Almost everyone who has programmed for language students has worked on a version of this pencil-and-paper game, and much of the work goes into producing exciting graphic displays culminating in gory deaths. It almost seems like an encouragement to the user to get the word wrong.

But can the computer version of HANGMAN offer anything that the pencil-and-paper version can't? If it comes with an authoring facility, each teacher can enter a personal choice of vocabulary very easily, perhaps reflecting words introduced during the previous week's lessons, or based on one or two particular spelling points. One attractive variation appears in the same collection of programs as the alphabet-trainer ABSHORT. In this version, two sets of data are stored — the words, and their definitions. The player is told that all the words in the data-store have certain letters in common (e.g. OUGH in cough, though, through, bought, etc.), and as each new word is selected at random from the data list, its definition appears on screen, and the user has to guess the word associated with that definition, and then spell it out, giving the letters *in the order in which they appear in the word.* This version clearly has more in common with a spelling program than with the more traditional HANGMAN guessing-game.

Some of the newer versions of the game intended for primary and/or secondary education (e.g. WORD SKILL) use the same word-game technique while abandoning HANGMAN's association with violence.

The game of WORDS gives students a chance to experiment with the behaviour of letters in English spelling. Two players (or, for preference, two pairs or groups of players) compete to see who can be first to build a word from letters offered in various ways by the computer. Since letters are offered at random, players who are more flexible and grasp opportunities to use what is offered, rather than holding out for the word they first thought of, are generally the ones to win. At any stage of the game, players can call up an on-screen "HELP" display showing how letters may be used initially, medially and finally in words, alone and in common combinations with other letters, so WORDS can be a game of strategy, not a hit-and-miss guessing-game.

WORDSPIN is described by its author as a "spelling trainer". The player has to identify a word which appears "spinning" on the screen. For example, the word "PLAYER" would appear successively as "LAYERP", "AYERPL", "YERPLA", "ERPLAY", "RPLAYE", "PLAYER" etc., until the player could identify it and "trap" the word in its correct form by pressing a key. Once the word has been "trapped", letters are blotted out, one at a time at first, then in chains of two, three or four letters, and the player types in the missing letter(s) — or, as the author puts it, "bombs the letters into the word". Finally, all the letters disappear and the whole word must be typed from memory. The version of the game listed by John Higgins and Tim Johns in their book *Computers in Language Learning* (Higgins & Johns, 1984) provides for word-sets to be stored, so the user can choose a set on which to work, rather than being presented with individual items of vocabulary with no particular linking theme.

Another type of spelling game allows words to drop randomly from the top of the screen, the player's job being to catch them before they reach the bottom. Two different keys are used, one to catch correctly-spelled words, the other for wrong spellings, and the computer logs the time taken to respond, so there is always an incentive to improve recognition of, and reaction to, the spellings.

SPELLERS attempts the same kind of thing, but with words or phrases (entered by the teacher via an authoring system) sailing from left to right across the screen, with the user "catching" the wrong spellings, while letting the correct ones sail by. A potential "plus" in SPELLERS is that the target words can form part of a phrase or sentence (which allows word-pairs like "their/there" to be used in context); a definite "minus" is that the only intimation to the user of success or failure is the adjustment of the score, which is not prominent on the screen, so that by the time the user has checked on the success or failure of one choice, the next item has sailed by unseen.

Word-games written with young children in mind often contain useful ideas, unfortunately marred for adult students by their child-oriented displays. SPELLERS, described above, is intended for children, but its screen display of a sunny landscape would be largely acceptable to an adult audience. HIGHFLIER also tests homophones, but in this case a correct choice leads to progress in an "aeroplane journey" between London and Paris, and the playing of a portion of "The Blue Danube". Teachers of adults have to make a difficult decision when purchasing such materials, balancing the usefulness of the concept against the degree of acceptability of the content, bearing in mind the adult audience for whom the program is intended.

Too much of a bad thing?

One of the dangers of a surfeit of word-identification games is expressed in *Computers and Reading Instruction* (Geoffrion, 1983). The authors are writing about programs for young children, though the warning could well be applied to some word-manipulation programs for adult language-learners:

> Aside from the reinforcement of correct responses, the computer contributes little that is unavailable in printed form at substantially lower cost. Moreover, these programs teach word identification as though it consisted of little more than memorising arbitrary letter strings, much like learning a set of telephone numbers. They ignore the rich meaning of words in natural language and overlook important cues such as context and semantics.

The authors then go on to describe a spelling-lesson of the future, in which all the words are seen in context, and the user is offered help of varying kinds, including:

- the spoken word (asking a speech-synthesiser to pronounce the chosen word correctly);
- division of the selected word into syllables to assist with pronunciation;
- the addition of the word to a personal dictionary of the student's "new acquisitions";
- pointers to "clues" to the meaning of the word from its context;
- display of the word's definition, if the clues fail.

Even the first of these options, which might appear to be the most problematic, is already in use in HONEYTALKER (a device developed by the MEP (Microelectronics Education Programme) for children with a word-recognition disability), which will read aloud any word which the child finds difficult when reading an on-screen text. The remaining options are well within the capabilities of the computers of today, though finding a currently-available program which makes use of any one of these features would be a difficult task.

3. Word-based Programs

Multiple-choice testing

Probably the earliest form of "educational" computer program was the multiple-choice test.

Multiple-choice abounds in computer software across the curriculum. It is easy to program, and once a suitable pattern is established for one test, a simple change of data makes a "new" test without further ado. Much of the home-computer "educational" software for O- and A-level practice falls into this category — ready-made tests on geography, English literature or whatever, all working on the same principle of one correct answer randomly placed among three distractors.

A typical example of a multiple-choice authoring-system designed for English-language teaching applications is CHOICEMASTER, which allows up to 30 questions to be entered in one test, using questions up to three lines in length.

A good multiple-choice authoring system should offer most of the following facilities.

(a) choices for the teacher, within the authoring system:

☐ free choice of the number of questions in each test, up to a maximum of, say, 40 (in EFL terms, enough to construct an RSA-type Objective Test);

☐ enough space to accommodate long questions and answers — say, up to three lines in a question, and up to one line for each of the answer-choices;

☐ the facility to specify whether questions should be presented in random order, or in a specific order (using the RSA Objective Test as an example again, it would be unfortunate to present the questions in random order, since the questions and answers together tell a story);

☐ an editing facility, with freedom to remove, add to or alter questions without having to re-type the whole test;

☐ the facility to spread the typing-in of data over more than one typing-session, if required;

☐ free choice of how many questions to include in each "run" of a randomised test — for example, a test may include 40 questions, of which the student sees only 10 questions chosen at random in each "run", so that the program may be used at least three times without the user noticing the repetition of questions;

☐ the facility to vary the instructions to the user according to the level of the test;

☐ choice of whether or not to offer totalling at the end of the "run" (e.g. "You have answered ten questions, and made eight mistakes");

☐ an option to include "help" pages or comments, if required;

☐ an optional branching mechanism to present further examples of particular types of questions, if required;

☐ selection of the number of distractors to be used in the test — usually a choice between A-B-C-D or A-B-C-D-E.

☐ the option of a print-out of a complete test, either for the teacher's own records, or for the production of class-sets;

(b) choices for the students:

☐ a menu of the tests present on the disc, preferably with comprehensible names — for example, "London Quiz No. 2", rather than "LONDQZ2";

☐ the "reconstitution" of cloze-type sentences, so that the user is not only told which choice is correct, but actually *sees* the correct answer installed in the sentence;

☐ the ability to leave the test gracefully at any point, without being obliged either to finish all the questions or "pull the plug" on the machine;

☐ the option of taking a printed copy of any question or questions required.

It would be a pity if multiple-choice ever became the principle use of the micro in language-learning. However, many students — in particular, those who need paper qualifications to "prove" themselves for various reasons — need to be able to answer multiple-choice questions, and the computer provides a relatively painless medium for practising the technique. This is especially so where sensible "help" pages are provided, so that the computer not only specifies the "correct" response, but explains *why* it is so, or points the student in the direction of further explanation or more detailed information, perhaps in a textbook. The "help" facility is one area in which the use of the micro can score over the use of a textbook for multiple-choice work. At least one multiple-choice program allows students to choose between a "tutorial" mode, with clues, and a "testing" mode, in which correct answers are given at the end of the test.

It is true that at present students actually seem to enjoy doing multiple-choice tests on the computer. The danger of a surfeit of such tests is that, sooner or later, they will begin to regard the micro as just another boring way to do boring exercises; so teachers should be careful to provide this type of exercise only in moderation, heavily diluted with programs which demand a greater degree of flexibility, imagination and intercommunication on the part of the users.

"Context-free" question-and-answer

A step up from straight multiple-choice is the "context-free" type of program, in which the user is presented with a question and is free to type in any suitable answer, which the computer then checks against its "bank" of permitted answers for that question.

Typical examples are the TEACHER'S TOOLKIT and the Tandberg TES/T program, both of which were intended for across-the-curriculum ("context-free") applications.

The basis of this type of program is the computer's ability to compare one answer with others, often selecting its next course of action on the basis of the results of the comparison. An example of the basic question-and-answer procedure in the TEACHER'S TOOLKIT is:

Computer: In what year did World War II break out?

Learner: 1941

Computer: No, that's not what I want.
It was in the late 1930s.
Try again!

Learner: 1938

Computer: No, the answer is 1939.

The choice of answer here is fairly limited — it is either 1939 or it isn't — but even using this simple example, there is an outside chance that the user might type "nineteen thirty-nine" instead of "1939". Packages like the TEACHER'S TOOLKIT, therefore, must accommodate a considerable degree of complexity in the response.

If the question happened to be "Which English king had six wives?", the "correct" answer might be one of the following:

Henry the Eighth
HENRY THE EIGHTH
Henry viii
HENRY VIII
Henry 8th

— or indeed any one of a variety of others, including spelling errors, all of which would be perceived by a human being as "the same", but which the computer would see as "different" answers, so the program must be able to:

☐ overlook the distinction between upper and lower case, except where it is necessary to preserve it (i.e. it must accept "Tuesday" and "TUESDAY" as correct, but for language-learning purposes should probably reject "tuesday");

☐ overlook non-critical spaces, but preserve vital ones (e.g. "Sunday" would be correct, while "Sun Day" might be unacceptable);

☐ be able to recognise "nearly-right" responses, by a process of matching word-patterns and/or keywords in the answer;

☐ offer advice on receipt of a wrong answer.

In addition, it should also offer most of the relevant features listed in the section on multiple-choice tests.

Computers are not intelligent. And computer programs are only as intelligent as the programmer programs them to be. It follows therefore, that where TOOLKIT-type programs are concerned, the onus is on the teacher who keys in the questions to foresee and forestall student errors by making adequate specifications during the authoring process.

This is of particular importance when the program is intended for students who are already struggling to express themselves in a written language with which they are not thoroughly familiar. For one thing, students are likely to produce responses which you never anticipated in your wildest dreams, but which are for all practical purposes "correct". Again, if the program-builder fails to specify tolerance of spaces, or distinction between cases, a "right" answer can be marked as "wrong". In either case, the result is frustrating and incomprehensible to the user, who quickly tires of the program and may well condemn all computer-programs out of hand as a result. It follows that this type of authoring system is one which must be used with care, given painstaking trials with students of varying backgrounds or nationalities, and fine-tuned to cope with as many operating-problems as possible.

An Introduction to Computer-Assisted Language Teaching (Kenning, 1983) goes into the subject of response-matching in some depth, and although the book is intended for would-be language programmers, any teacher interested in making use of a package such as TEACHER'S TOOLKIT would be well advised to read this book first, as it gives a good insight into the kind of problems which will arise, and the reasons for them.

Created with care, such a question-and-answer test offers more flexibility than straight multiple-choice, usually with better opportunities for branching and reinforcement, and a chance of slightly more creativity on the part of the student.

Multiple choice for specific purposes

PHRASE and PHRASAL VERBS are examples of multiple-choice programs restricted to a specific purpose. In the case of PHRASE, the idea is to practise phrases or idioms based on a theme (for example, "colours" or "animals"); PHRASAL VERBS uses a similar format to present phrasal-verb combinations. Both programs involve the indentification of phrases and their meanings, and include contextualised examples.

Both are ordinary multiple-choice-type programs, but they contain one feature worthy of comment. Many ready-made commercial multiple-choice programs use a "pool" of data from which to draw the distractors at random, so that on many occasions choosing the correct answer can mean simply weeding out the three really silly choices and seeing what you are left with. PHRASE and PHRASAL VERBS, and any multiple-choice authoring system worth its salt, provide each question with its own set of credible distractors, which means that the teacher responsible for entering data can offer students a genuine choice of answers.

A variation on the PHRASE theme is described by Jeremy Fox, in *Computer-assisted vocabulary learning* (Fox, 1984):

> On the screen, after the students have been given initial instructions, it looks like this:

```
(TYPE H FOR HELP, OR X TO EXIT FROM THE PROGRAM.)
NOUNS:       loss        arts         law
             load        rags         tidings
             lorry       certainty    crisis
ADJECTIVE:   dead
```

The student then has to pick the two nouns with which the adjective "dead" most easily collocates. If they get stuck, they can type "H" for "Help", and are given clues in a mini-context, usually a sentence:

It's a dead......................................he'll come.

This pen's a dead....................................... It won't write."

Many word-based programs are, like the above, lightly-disguised multiple-choice tests, but are none the less useful for that.

Hidden-word games

A whole genre of word-games involves sorting out words hidden in various ways from the player, or finding words from jumbled letters. Some of these programs have "help" facilities in the form of clues or contextualisation, and most allow new words and clues to be added to the original lists supplied with the program.

An excellent version of SCRABBLE is now available for several makes of computer — the advantage (for language learners) over "real" Scrabble being that the computer has its own dictionary (8,000-11,000 words depending on the computer being used) and challenges any words which it doesn't know. A fascinating aspect of computer SCRABBLE is that players can watch the computer "think about" its moves, as it tries out different words in order to maximise its score. The computer can be set to play at any one of four levels, so that it need not always win.

Other programs, like CROSSWORD CHALLENGE, offer more or less straight crosswords, played on the computer. CROSSWORD MASTER is an authoring system for crossword-minded teachers.

There are various WORDSQUARE-type programs, involving words hidden in various directions in a grid of letters; most programs allow teachers to "author" their own vocabulary; most simply provide an on-screen word-list like the traditional wordsquare puzzles, some offer clues instead of the words themselves. Some of these programs allow students to play directly on the computer, or offer the option of printing out the game, either an individual copy, or directly on to a Banda so that a class-set may be duplicated.

Two programs intended only for the production of paper-copies (that is, there is no way of playing the games on the computer) are MASK and SHUFFLE. Each comes with a dozen or so ready-made puzzles, plus an authoring facility so that more puzzles can be added.

4. Text-based programs

Branching stories

One of the most obviously "text-based" program-types is the branching story or maze, which leads the reader through a text, presenting it in manageable chunks, until by one means or another the end of the story is reached — or, in some cases, one of several "ends" of the story.

There are two kinds of branching story, and although the second type is more interesting than the first, the first nevertheless has a useful application in terms of comprehension work.

The first type is a "straight" story. There is only one way of getting through to the end, and that is the "right" way. En route, the reader must answer multiple-choice questions about the text. If the answer is wrong, there is an explanation of why it is wrong and the reader sent back to choose again. If it is right, there is an explanation of why it is right, and the story continues.

It is possible to do the same exercise from a book, but you cannot cheat the computer by peeping at the answers, and the computer also does the page-turning for you, which is useful in a long text where there is much to-ing and fro-ing from page to page according to the answers you give.

In this kind of branching story, as there is only one route to take, the questions must involve comprehension of the text of the story so far, rather than offering real choices about what to do next. Has the reader understood the implications of certain words in the story? In view of what has happened so far, which of these alternatives is likely to happen next? And there is only one "correct" answer. In fact, it is rather like a multiple-choice comprehension passage, and that is exactly how it can be used. A simple branching-story authoring program is an ideal way of putting over comprehension techniques, because it not only leads inevitably to the "correct" answers, but explains in doing so the reasons *why* the other choices are incorrect.

Students can work through reading comprehension passages in books. But the book will only say *which* answer is correct, not *why*, and that is the scoring feature of the computer — each multiple-choice answer can have its own "page" of explanation, after which the reader progresses to the next question, or loops back to have another go at finding the right answer. This kind of program can be useful in helping students to acquire *interpretive* reading skills — the ability to read between the lines — rather than simply reading for information. As far as the teacher is concerned, entering a new text is simply a question of typing the text itself along with its related "pages" of explanation — the authoring program organises everything else.

The second type of branching story, or maze, is far more entertaining. There is no fixed path through the story, so the reader can be invited to make real (multiple-) choices on the lines of "what would you do next?", and the story goes off in different directions according to the choice that is made. This is much more fun as a group activity, where students can discuss pros and cons before making a choice. It is also more economical, as the same group can tackle the same story several times, and take a different path on each occasion. Another option in such a maze might be a "blank page" feature, allowing students to make up their own story at certain points, and add it to the maze.

Some of these mazes are small and relatively simple. They need not be complex in structure in order to achieve their purpose of provoking thought and discussion amongst a group of users; in fact, for adult students, the subject matter is infinitely more important that the length or complexity. Mazes can rival adventure games in their size, structure and subject matter (an example of this type is THE DARK CASTLE, written at ILECC for French learners), but a smaller maze which invites progress through a series of social or moral dilemmas is likely to have a wider appeal amongst adults, and will certainly be far more useful in improving everyday language skills. Programs such as BRANCHING STORY and STORYWRITER are "maze generators" which allow teachers and/or students to determine the content of the maze for themselves.

Reading programs

There are many ways of improving reading skills using a micro. SPEED uses the simple technique of presenting text a line at a time, with a "bouncing ball" marker over the words, to encourage regularity of reading-speed. The reader chooses in advance the required speed, on a scale from 1 to 9, which accommodates a wide range of ability. Text can be entered via an authoring system. It is the "author's" duty to ensure that line-breaks occur at suitable places — not simply so that words do not run over into the following line, but in order that a "line" of text may present a meaningful unit to the reader, since only one line is seen on the screen at any one time. The sentence "Jack had a job as a porter in the local hospital, where he had been working for five years" would appear on the screen as three disjointed lines as follows:

Line 1: Jack had a job as a porter in the local
Line 2: hospital, where he had been working for
Line 3: five years.

This could well be offputting for a novice reader, and it would be better to introduce extra spaces into the text, so that it would read:

Line 1: Jack had a job as a porter
Line 2: in the local hospital,
Line 3: where he had been working
Line 4: for five years.

ELREAD offers phrases rather than a complete text. There is an authoring program so that teachers can generate collections of phrases based on themes such as "silent letters" or "abbreviations". The object is the improvement of powers of observation — apostrophes, capital letters, question-marks, the correct form of typing the date, and so on. The program presents a phrase (the initial display period ranges from 2 to 9 seconds, chosen by the user) on the screen. When the phrase has disappeared, the reader types it from the keyboard, attempting to reproduce exactly what was on the screen. The program picks up any differences, pointing out and offering corrections of missing or extra letters, metathesis, and case errors. If desperate (that is, the user's version differs considerably from the computer's version), the program resorts to asking the user to re-type part of the phrase. The speed of presentation depends on the user's performance — correct responses speed up the program, errors slow it down.

MIKE AND ANNA attaches reading-speed to meaning, in a way which is likely to catch the reader's imagination. Some information about the characters Mike and Anna appears in the top portion of the screen. The computer then generates random sentences about Mike and Anna, which may or may not be "true". These sentences travel from right to left across the bottom of the screen, and the reader has to try to "catch" the ones which give false information before they slip away. The level is simple — a typical sentence might be "Anna is in London" — but very stimulating.

Jumblers

There are various versions of "jumblers", which require a close understanding of the text, but which do not emphasise speed of response so much as intelligent manipulation. Probably the classics in this field are Tim Johns' JUMBLEWORD, JUMBLECHUNK and JUMBLEPARAGRAPH. JUMBLEWORD requires the reorganisation of an anagram. JUMBLECHUNK involves the reordering of words within a clause. JUMBLEPARAGRAPH asks for the reordering of sentences in a text. All three versions present the material in the context of a complete story or text. Tim Johns and John Higgins explain, in their paper *Approaches to CALL for English as a Foreign Language: A summary of work in progress* (Higgins & Johns, 1983):

> One of our goals is to develop a suite of such programs which can operate on a series of randomly-accessed texts stored on discs at different levels of difficulty and of different types (ranging from children's stories to scientific reports). This would allow the student great flexibility in choosing the sort of text he wishes to work with and the game/activity he wishes to use: such flexibility is, we believe, of particular value to learners of English (or, indeed, any other language) for Specific Purposes. A well-designed program of this type can be extremely effective in concentrating the student's attention on the form, meaning and "connectivity" of text and developing the crucial skill of successful guessing.

It is quite easy to type out a text on paper, cut it into pieces, and give the pieces to a group of students to play with. But even the most ardent anti-computerist would find it difficult to argue that this is better than, or even as good as, a computer version where:

- ☐ any number of texts can be available to the students, just for the price of typing them in at the keyboard;
- ☐ words, sentences and paragraphs can be arranged and rearranged on the screen, always appearing in the natural form of continuous text rather than as disconnected bits of paper on a desktop;
- ☐ a variety of different activities can be generated, according to the personal choice of the users, without any special preparation or intervention by the teacher;
- ☐ the activities can develop from the detailed (letter-jumbling) to the general (paragraph-jumbling), using the same text as base material;
- ☐ there are no paper pieces to be lost!

Very often, a potentially useful text is almost wasted on students, because they feel that, having read it once, they "know" it, and have no interest in investigating its hidden depths. Programs like the jumblers encourage readers to dig deep into a piece of reading material, and extract from it every ounce of benefit in terms of structure, spelling, vocabulary and meaning.

Cloze programs

Cloze-testing is part of the time-honoured tradition of language-practice, and followed close on the heels of multiple-choice in the computerisation stakes. READAMATICS and CLOZEMASTER are typical examples, the former intended for schoolchildren, the latter for EFL adults. Cloze programs should offer most of the following features:

- ☐ ability to delete every Nth word, or alternatively words of the teacher's choice;
- ☐ ability to enter a variety of credible alternatives if desired;
- ☐ ability to print out a Clozed text;
- ☐ facility for disguising the length of Clozed words by replacing with a uniform number of dashes, or alternatively give a clue by leaving dashes representing the word-length;
- ☐ ability of the program to recognise "nearly right" answers and respond accordingly (READAMATICS, for example, detects spelling errors and provides a template of the correct answer to assist the reader in making another attempt).

One feature to look out for particularly, as it might strongly affect one's opinion of a Cloze-making program, is the second of the list above: the ability to enter credible alternatives to the Clozed words.

Some programs are deliberately student-oriented. They are intended for an adult audience, and give complete power to the adult user to manipulate texts as required. This means that, although the teacher decides which texts to enter, it is the student who decides the word-frequency of the Clozeing. It is clearly impossible for every word in every text to be assigned "credible alternatives", and the result is that this kind of program will accept *only* the word in the original text to fill the gap. Teacher-oriented programs allow the teacher to specify which words are to be Clozed, and also to specify which alternatives are permissible.

Both types of program are flexible, but they are flexible in different ways. The student-oriented type allows the user to explore the same text several times by specifying different deletion-frequencies. Inevitably, the educational objectives of this kind of Cloze will be less clear than when the teacher specifies the deletion; on the other hand, the opportunity for examining each text in depth is much greater. The teacher-oriented type permits only one version of deletion to be seen by the user, and that version is not necessarily "genuine" Cloze (it may have an irregular deletion pattern); on the plus side, however, the teacher may specify deletion of words of a particular kind (all prepositions, all verbs, etc.), and may additionally allow for a greater degree of imagination on the part of the user (accepting "leapt" and "sprang" as alternatives to "jumped", for example).

The ultimate Cloze test

The ultimate in gap-filling exercises is the type in which *all* the words are removed from a text, and the player, starting from nothing, attempts to rebuild the text (and accumulate a large score at the same time). DEVELOPING TRAY, STORYBOARD, TEXTBAG and TEXTBUILDER are examples of this type. Different versions offer varying facilities, but among the choices offered are:

(a) for the teacher:

☐ an authoring facility to enable new texts to be entered and saved;

☐ a facility to allow the editing of existing texts (this is important when the texts may be several pages in length, less important for a single-page text which can easily be re-typed);

(b) for the players

☐ a menu of text-titles to choose from;

☐ an option to preview the text if required;

☐ a choice of

  (i) removal of all text, leaving no clues for the players;
  (ii) replacement of text by dashes or asterisks, so that word-boundaries may be seen;
  (iii) removal of all alphabetic characters, leaving punctuation-marks as clues;
  (iv) replacement of certain letters as a "starter';

The players can move a pointer freely over the screen, and their task is to position the pointer and then try to predict letters or words in that position, which involves knowledge of structure, spelling, and — as the text becomes more complete — meaning. Help can usually be sought, often in the form of "buying" letters (all the incidences of the letter you "buy" appear in the appropriate places on the screen, and your score is debited appropriately) or "buying" whole words (you "buy" the word on which you have placed the screen pointer).

There are now many versions of this type of program, some written for school use, others with Engligh language learners in mind, so this is one program where genuine comparisons can be made as to the suitability of the different versions for an audience of language-learners. There are a number of factors to look out for:

- ☐ some versions allow texts of several pages in length to be entered, sometimes at the expense of other features in the program, while others limit a text to a single screen-page;
- ☐ some versions only allow computer-type names for texts (for example, "PRANDPR" instead of "PRIDE AND PREJUDICE"), while others allow long titles to be used;
- ☐ some versions use "real English" prompts, others go for "condensed English" in order to save space;
- ☐ some versions will not reveal the whole text unless the user plays the game to the bitter end — regardless of the constraints of time and sheer frustration; others allow players to exit at any stage;
- ☐ some versions offer, alongside the text itself, a series of questions through which the players can not only increase their score but also check their understanding of the text as it is revealed. Questions can also focus attention on important parts of the text.

Because of the wide choice now available, it is worth checking up on the facilities offered by various versions, to see which best suits the target audience. The only essential feature is that teachers should be able to enter *their own choice of texts*, so that the level of language is appropriate to the students for whom the texts are intended.

These programs are among the most stimulating yet on offer for English language learners, and certainly the most enthusiastically received by students and teachers alike. The possibilities for texts are endless: a collection of extracts from the classics, a collection of passages based on reading materials used in other classroom activities, a collection featuring other subject-areas which the students may be studying in addition to English; a set tailor-made for a particular student with a specific problem in spelling or word-recognition.

Although the DEVELOPING TRAY-type programs are not competitive in the sense of one player "against" another, they provide an opportunity for players to pit their wits against the English language. And this type of program certainly works best when several students pool their ideas and build the text together, as this stimulates much discussion as to the probabilities and possibilities of various options.

5. Database and rule programs

The value of a computer lies in its ability to store information and manipulate it at a phenomenal rate, and to do it again and again, strictly according to the rules imposed by the programmer.

In the early days of computing, this ability was used only in handling large quantities of numbers and performing complex arithmetical tasks, but it can be equally well exploited in language-based programs. Two ways of using these capabilities of the micro are by providing it with stores of information which are available to the user on demand — that is, setting up a database — or by programming it to recognise some of the rules of language, so that when it is "questioned" by the user it can come up with the right answers — most of the time, at least.

Electronic magazines

The use of general-purpose databases in the classroom will be discussed in Chapter 2. This section deals only with specialised databases designed to fulfill a particular function. One increasingly popular kind of database is the "electronic magazine", which is a miniature version of the teletext service (like Ceefax or Oracle), in which students (or teachers) can set up "pages" of information which can then be accessed by other users in the same "closed circuit" system.

Paul Power, in his article *The Micro Way* (Power, 1984), writes:

> Using a database the teacher is able to format and present to the students an array of information. It would be possible to build up a profile of details of shops, cinemas, restaurants, etc., within an area. It is possible, for example, to have students work out a mini teletext system or to design an electronic magazine or newspaper.

A few micro-users have already set up "electronic magazines". The Bell College (EFL) at Saffron Walden has such a system on its network of BBC machines, and this is written and managed by volunteer student contributors. In general education, the Stantonbury Campus in Milton Keynes uses its network as an electronic noticeboard (Fluck, 1984), as well as in a large variety of other educational and administrative ways.

This use of a specialised database program to make a public noticeboard is a worthwhile long-term project, but many other databases come in a ready-to-use form, or can be set up with a minimal amount of effort.

Databases for language work

Simplified databases, specifically geared to language work, may be used either as programs in their own right, such as HOBBIES, or as an incidental part of a program, as in PERSONAL, where the database can be referred to for vocabulary assistance if needed.

PERSONAL is based on an idea borrowed from the field of psychology, and is an example of the very simplest form of database, which is little more than a word-list. The program calculates the user's "honest" opinion of five friends, using five of their good qualities as a basis for the calculation. This pseudo-psychological piece of fun needs to be fed with data, and good qualities may not spring readily to mind, so a miniature database of seventy qualities, ranging from "broadmindedness" and "concern for others" to "self-control" and "a sense of responsibility", can be consulted for reference purposes. The qualities are presented in random order, and the effect is rather like examining a tiny thesaurus.

The HOBBIES program addresses itself, via a database, to a very real need amongst students especially those living away from home. Many students find themselves at a loose end, either because they have no leisure interests at all, or because they have no idea how to go about pursuing their existing interests in a new environment. HOBBIES offers two "services" — it can take you directly to an existing hobby and offer information about what facilities are available in the area, or, by judicious questioning, it can suggest a list of suitable hobbies, any of which may be followed up in more detail.

The program as supplied contains no detailed information. This information must be gathered locally and entered via the authoring program. Students might be encouraged to go out hobby-hunting, to assemble relevant details. Up to a screen-page of information is available for every hobby, with simple commands to format the page as required. The system is simple to use, and new pages can be entered by students or staff (although much of the pleasure of setting up the system would be obtained from assisting students to do it themselves).

Once entered, the information is available to any interested person, and a printed copy of the hobby-pages can be obtained, of which an example appears on the following page.

One of the better-known database programs actually contains no data at all, or very little, and this is ANIMAL, of which there are many versions. The value of ANIMAL for language students is in the construction of closed questions. The program "learns" about animals (and thus builds its database afresh each time it is used) from the students who use it, who have to supply appropriate questions and answers to help the computer to distinguish between animals of various types. This obviously requires students to learn something about animal-based vocabulary — which may be as elementary as "wings", "tail" or "trunk" — but it also demands an intimate knowledge of question-forms, such as:

Has it got a long neck?   Can it fly?   Does it eat people?

```
SWIMMING

The nearest swimming baths are at:

          Allsports Sports Centre,
          Leadown Road, N.W.3.
          (near Safe-o Supermarket)

To contact the local swimming club, write to:

          The Secretary,
          Chapelforth Swimming Club,
          Wilton Road, N.W.6.

You can get swimming lessons at the Sports Centre. The cost is
about three pounds an hour. There is also sub-aqua instruction.

"Fitsports", 120 Leadown Road, offer a 10% discount on swimming
equipment to students with up-to-date student cards. Remember to
take your current student card, and say you are a Northwest
College student.

If you have any questions about swimming, ask Jane Smith
(Room 12).
```

Output from HOBBIES program

Most programs may be used without the supervision or intervention of a teacher, but this is one which needs vetting during its use. The program will not jib at accepting "Have it got long neck?" as a reasonable question. Thus ANIMAL is one of the few programs which actually works better as a whole-class, supervised activity, not least because a large group will provide a vast range of possible questions, which may be discussed by the class before a final decision is made on which is most suitable. Some versions of ANIMAL are not limited to animals, but provide a variety of subject-areas to explore. The "animals" topic is simple enough to be used by students at any level of competence, but some of the other subjects can get very sophisticated.

JACKASS is a version which allows a choice of Animals, Objects, Cities and People, and which incorporates the useful facility of a "report phase", when the computer produces a piece of connected prose, describing the facts which it has been "taught" by the users.

In the process they learned to frame questions, capable of yes/no answers, to distinguish between two animals (or indeed two anythings). They spent an afternoon writing questions which might be useful for playing ANIMAL.

The above extract is from an *ILEA Computers in Primary Schools* Newsletter (Fraser, 1984) and "they" were children at an infant school. They might equally well have been adult students. (The infants followed up their investigations with a visit to the zoo.)

SEEK, a program similar to ANIMAL, was used by students classifying types of cheese in an example quoted by Jan Stewart (Stewart, 1983) in *Exploring English with Microcomputers.* The participants were provided with labelled samples of cheese, and were invited to "teach" the program how to distinguish between the varieties on the basis of taste, texture and appearance, which suggests the possibility of using other realia to encourage questions based on the senses — "Does it sound harsh?", "Has it got a salty taste?", and so on.

Databases need to be fed with information. In a few cases, the information is supplied with the program (as in PERSONAL), but mostly the user is required to assemble and enter the relevant details. That is a large part of the value of a database in language-work; students can question it and take information *out*, but they can also do the much more mind-stretching work of getting the information together and putting it *in*, which could involve questioning or consulting people, and looking up details in newspapers or in an encyclopaedia, as well as discussion and co-operation in sifting the information and deciding what to use and what to leave out.

Rule-based programs

A second way of using the computer's manipulative capability is by feeding it with rules about language and then getting the user to question it to find out how much it knows.

One of the classics in this field is LOAN, in which the computer offers to borrow a sum of money for the user. It needs to know how much money you need, and who you want to borrow from, after which it supplies a suitable request — in the words of the author

> ranging from 'Got 10p on you?' (for a member of the family) to 'You couldn't possibly see your way to letting me have an overdraft of £5,000, could you?' (for a bank manager). (Johns, 1982a)

By examining LOAN's suggestions for borrowing cash, much can be learned about formal and informal requests, which may be applied to things other than money, and the program could well be used as the basis for a discussion of register.

Two other programs from the same authors (Tim Johns and John Higgins) are tiny enough to fit into the memory of the smallest computer, and were originally written for the 1K-version of the ZX81. The concepts behind these two programs (one on s-endings and one on the indefinite article) are so important that it is worth quoting Tim Johns' explanation of them at some length, from his paper *Exploratory CAL* (Johns, 1982b), which also contains listings for the two programs:

> In place of the long and complex program required for traditional CAL, we have short and apparently trivial programs which take no more than an hour or two to write . . . The relationship of the student to the computer program is also different. In traditional CAL the program that controls the teaching is normally a matter of indifference to the student . . . inspecting the program could hardly be of much use in language-learning. With exploratory CAL, however, it is a test of the effectiveness of a program that at some stage the students should demand to see it, to 'find out how the trick is done'. The program becomes a part of the language-learning process as a (partial) explanation of how the target language operates . . .
>
> (The s-endings program) offers to accept any 'English' word, whether real or invented, and to add an 's' to form the third person of a verb, or the plural of a noun. The way I have used 's-ending' is to give a group of students five to ten minutes to test out as many words as they want on the program, keeping a record of its 'hits' and 'misses': we then discuss the rules the computer has applied.

The students will try to catch the computer out by offering it examples of irregular plural formations, which seem to produce 'misses' for the computer, such as "SHEEPS" and "KNIFES", but even these can

> form the basis of some valuable later discussion: for example, by the terms of the computer's offer, KNIFE could be a verb, and KNIFES therefore correct.

The value of this program clearly centres on students finding out "how the computer does it", and then, having seen that s-endings depend upon the application of a very few readily-stated rules, absorbing the rules into their personal "database" of language-manipulation techniques. The second example, also contained within twenty or so lines of programming, will match the appropriate form of indefinite article with a given phrase (it will cope with "a uniformed person" and "an uninformed person", "an 11-metre yacht" and "a 110 film", for example). But, as Tim Johns continues, this type of program

> makes demands on the student, requiring an alert and intelligent attitude to the task of learning. The writer is fortunate in that his own students (overseas postgraduates at a UK university) take well, on the whole, to the approach. With slow learners, however, exploratory CAL may run the risk of mystifying and confusing the student if it is not carefully prepared for and integrated into the overall programme.

With the right students, and a teacher present to help with the computer's 'misses', this type of program may be helpful in persuading students to enquire into the terms on which language operates. (They may also perform a useful incidental function by demonstrating that computers are not infallible!) They are not, however, the sort of programs on which students may be safely let loose on a self-access basis!

6. Structure-based programs

Mainly of interest to the EFL sector, structure-based programs are few in number. Hardly any programs attempt to present structure; nearly all are content to allow the user to practise what has been learned elsewhere.

TWO STICKS works in the same way as MIKE AND ANNA, with an added ingredient — comparison of adjectives. Two "sticks" of random lengths chosen by the computer are drawn on the screen. The program generates sentences comparing the two sticks, and the user decides whether or not the statements are correct. Starting with basic comparisons ("A is as long as B"), the user progresses to more complex relationships ("B is marginally longer than A").

PHOTOFIT also uses comparison as its theme. A face is drawn on the screen, the student examines it, and the face then disappears. The student has to recreate the face by giving instructions to the machine, asking it to draw the features, and modifying them if necessary by asking for the hair to be drawn longer, the eyes bigger and so on, until the new face seems to be identical to the original one. The original face then appears alongside the new one, and a list of differences between the two is generated.

THE CRANE GAME — described in *Computers and Reading Instruction* (Geoffrion, 1983), and probably based on Winograd's artificial intelligence experiment, SHRDLU — is a program-idea featuring the positional relationships of objects moved about on the screen by a "crane", in response to the user's instructions. "Put the red block under the green one" could be achieved by using a screen pointer to choose an object, a preposition, and a second object, and allowing the program to synthesise the sentence, print it on the screen, and match the crane's actions to the words. Alternatively, the user could type in the complete instruction, to be analysed and acted upon by the program, although this would require much more sophisticated programming techniques to cope with errors on the part of the user.

Other structure-type programs involve telling the time from an on-screen clockface, and, at a more advanced level, joining pairs of sentences using appropriate conjunctions.

The program LIFESPAN uses structure as an incidental by-product, the main object being to ask the user to read and respond to the kind of questions which may be asked in the doctor's surgery. The computer asks questions about the user's lifestyle (the amount of regular exercise taken, smoking and drinking habits, and so on) and then calculates the expected lifespan. This EFL-based version "remembers" the answers given by users, and makes suggestions, based on a dozen common patterns used in English, as to how their chances of survival may be improved. The advice ranges from "If I were you, I'd . . ." to "Have you ever considered", produced randomly by the program.

FORSINCE is in the programmed-learning mould — or at least, it contains an optional tutorial element, backed up by two fast-reaction "FOR or SINCE?" games, one basic, the other more difficult ("since the last time we met", "for the best part of two years"). The tutorial element, which is not intended as an introduction to the subject but as follow-up material, uses real-time examples, which vary according to data supplied by the user about the time, the day, the month and so on. The student therefore always sees the sample sentences in relation to the present time, which is vital to the understanding of the correct use of "for" and "since".

Structure-based programs are unlikely ever to become star attractions in CALL, although they can play a solid and worthy role as reinforcement materials — especially when students can be steered towards a relevant program after missing a lesson, or if a small group needs extra practice.

7. Adventure games

In strong contrast to structure-based programs, the adventure game gives a certain degree of freedom to the user — or users, since adventures lend themselves very much to group activity.

An adventure-game takes its players on a fantastic journey, such as exploring caverns inhabited by dangerous creatures, or seeking treasure in a magic castle. The players progress from one death-defying situation to the next by giving instructions to the program — for example "go north", "throw axe" or "take diamonds". The skill of adventuring is in discovering as quickly as possible what commands the program will understand, and what tricks the programmer has used to make life difficult for the adventurers.

The adventure is borrowed from mainstream home-computer games, and in some cases it is not only the idea which is borrowed but the whole adventure, unaltered. Many people find adventures so absorbing that they can spend months on one game, exploring the ramifications of dungeons and forests, collecting treasure, warding off evil creatures, and investigating the elliptical adventure-language. While this may be fun, it has a number of drawbacks when used as an educational tool, especially when the "adventurers" are people whose knowledge of spoken and/or written English is limited.

The chief drawbacks from an English-language point of view are the very attributes which make adventures a success in the home-computer market:

☐ the games are very long and complex, and the chances of a group of students working through a whole adventure are very remote;

☐ the subject-matter is not universally appealing, and certainly is of very little use in the everyday world — how often do you meet a gnome with an axe on the way home from the Jobcentre?

☐ the language-input required from the user is very specialised, and of a condensed nature peculiar to adventure games. The average commercial adventure game packs as much action as possible into the computer's memory, and leaves little room for handling language. The program's interpreter as a rule only understands two-word commands, which means that articles, prepositions and the like go by the board. "Take key" is a perfectly acceptable adventure-game command; but it is not acceptable spoken English.

On the other hand, adventure-game supporters argue that:

☐ the length of an adventure is irrelevant so long as students benefit from whatever small portion of it they are able to complete; in any case, games can often be "saved" on a disc or cassette, and a group can return at a later date to pick up where they left off;

☐ the subject-matter is irrelevant, since the object in the language-classroom is to get students communicating in the target language, and adventure games certainly promote discussion and group-work in terms of the next course of action to be taken; what is more, all the scene-setting is done in descriptive language on the screen, and this too requires comprehension and discussion.

☐ the abnormal language which the computer requires as a response from the players is a tiny proportion of the whole, and students are unlikely to confuse this with "real" language; in fact, the differences between computer-language and "real" language might even be exploited.

So there are two ways of looking at adventure games. Either they are a good way of getting a group of students talking and discussing strategy, and the linguistic limitations are unimportant. Or the language-problem is paramount, and all the strategy in the world won't help a student to understand that "take key" only works in computer-language, not in the real world.

There are teachers of both persuasions, and not only in adult education. A primary teacher, writing about his experience of adventures in an *ILEA Computers in Primary Schools* Newsletter (Duggan, 1984) said:

> The repetitive nature of the language involved meant that bilingual children and those with reading difficulties were very quickly able to take a full share in mapping, operating the keyboard and joining in the decision-making.

But another teacher (Wisely, 1984) wrote in the same newsletter:

> I began in an optimistic mood but am now very unconvinced. To be useful in the primary classroom an adventure/fantasy game would need to be very different from the one I have been looking at. It would need to be attractive graphically, have much less print on the screen at any one time, have a set of clues and help routines . . .

One important factor in the success of adventure games in the classroom will be the attitude of the teacher — a convinced adventure-phile will no doubt have greater success with adventure games in the classroom than an adventure-phobe.

COLOSSAL ADVENTURE and ADVENTURE are two typical large-scale adventure-games; THE HOBBIT is less typical since it accepts multiple and complex commands in its own language — "Inglish" — ("Take the key and unlock the door", and "Open all bottles except the green one", for example), allows you to "talk to" other characters "Say to Gandalf 'Read map'"), and endows the other characters with more or less realistic and sometimes volatile personalities.

Programs written specifically for the commercial market rather than the educational one often contain spelling or structural solecisms, which are not open to amendment. They may also contain "non-standard" dialogue, such as the following conversation between trolls in THE HOBBIT:

> "Look at this! Can yer cook 'em?"
> "Yer can try, but he wouldn't make above a mouthful."

A few adventure-style games have been written for the schools market. One adventure for children, GRANNY'S GARDEN, allows the players to make spelling errors, which are accepted without comment, which may be considered a point in its favour by some teachers. It is an important step forward in the sense that, unlike most programs, this one can interpret what the player is likely to have meant, rather than rejecting the whole input out of hand because it contains a spelling or typing mistake.

KNIGHT'S ADVENTURE is unusual, in that it comprises five sections, only one of which has a conventional adventure-type structure. The other sections involve problem-solving (not numerical) and riddles. It is much more compact than most adventures, but may still be difficult for learners of English.

THE DARK CASTLE (an ILECC program) is unfortunately written in French for secondary-level children. Unfortunately, because an English version of it would be a very useful introduction to adventures. It has one feature which could well be copied in any adventure-game written for English language learners, and that is its excellent dictionary. DARK CASTLE players have only to position a marker over an unfamiliar word and press a key, in order to see a translation of the word *in context* on the screen.

The ILECC programming team's solution to the language problem in THE DARK CASTLE, was to turn the adventure into a branching story, with multiple-choice options. This sidestepped difficulties with input since the only input required was in the form of multiple-choice answers. But half the fun of a real adventure is in exploring the program's vocabulary — thinking up new commands to try and persuade the locked chest to give up its gold, or to escape from the clutches of the evil giant.

8. Simulation and role-play

Perhaps the best-known kind of simulation is the flight-simulator. The flight-simulators used for pilot-training are huge in size and expense, and provide an experience that comes close to a real flight.

A flight-simulator embodies all that is easy to achieve in simulation-techniques, as well as everything that is difficult. The easy part is the handling of numbers. A flight-deck is provided, so the trainee pilot operates a full-scale model plane as if it were the real thing. Every action of the pilot is interpreted in terms of the simulator's internal (mathematical) model of how a real plane would behave in similar circumstances, and the results are fed back to the flight-deck and reflected in altered readings on the instrument-panel. Movement is accomplished hydraulically, and the pilot experiences the same sensations of movement as are produced in a real flight, although in fact the simulator is no more than a few metres from the ground.

The really difficult part is in making the pilot's environment as realistic as possible. Looking out of the "windows" of the simulator, the pilot sees a computer-generated "view", which can be so sophisticated that the choice of "a flight from Heathrow to Orly" will display exactly what would be seen during the real flight. The view is not only correct in detail (of the airport layouts, for example), but alters in perspective to match whatever action the pilot takes.

It is possible to buy, for a few pounds, flight-simulation programs for home-computers which are really simulations of flight-simulators. They provide a taste of what flying a plane is all about, with everything scaled down to fit on to a TV screen. Instead of real switches on a real flight-deck, the "pilot" presses keys on the keyboard, and the "dials" on the screen respond, rather more slowly that the real thing. Even the "view" from the cabin is simulated in some versions.

Simulation techniques using massive mainframe computers are employed in business, in government, in sport, in space exploration and many other fields, to answer the question "What would happen if..?" The same techniques can be employed on a small scale using a microcomputer, but prospective users should be aware that what they are buying is probably the "easy bits" of simulation (the numberhandling), the "difficult bits" being either cut down or left out altogether. As far as computers are concerned, language-handling is one of the difficult bits.

Some micro simulations

What the student types in at the keyboard is therefore likely to be even more circumscribed in a simulation than in an adventure-game, and simulations are likely to be based on subjects which lend themselves to numerical evaluation. The language value is therefore wholly in the discussion which the program generates about the situation being simulated.

Some of the simplest "fun" simulations depend on random elements to such an extent that whatever choices the players make, the computer seems to go its own way regardless, and when players can see that their actions do not affect the outcome to any great extent, they become bored and give up. On the other hand, "serious" simulations may well be about subjects which will interest a particular type of student but will not have a general appeal.

The current generation of simulations includes:

- STOCKMARKET, which involves buying and selling shares and becoming a millionaire in the process;
- STOKMARK, which is similar to STOCKMARKET;
- TELEMARK, in which the players run a company, and hope that it makes a profit;
- INHERITANCE, on similar lines to TELEMARK.

The numerical bias in these subjects is evident, and people who enjoy playing Monopoly will probably enjoy this type of simulation. More likely to attract the average student are:

- SURVIVOR, in which the players are shipwrecked and have to stay alive until rescued;
- SLICK! a BP production, giving an opportunity for potential conservationists to control an oil slick and minimise the damage it causes;
- FARM, which asks the "farmer" to grow and store enough grain to feed the family and random guests, while surviving the vagaries of the weather and potential disasters. This simulation offers useful practice in reading and interpreting bar-charts;
- GREAT BRITAIN LTD., which gives the players the chance of being Prime Minister, running the country for a while and then standing for re-election;
- SUBURBAN FOX, a chance to examine the problems faced by foxes in a city environment. The user plays the part of the fox, and attempts to survive various dangers in the form of road-traffic, dogs and so on, while finding enough food and water to keep alive.

Even these, though of less narrow an interest, have little relevance to daily life as most of us know it. Their main value to adult students lies in the communication which they generate within a group, which can be considerable in quantity and quality. The advantage of the computer simulation over a whole-class non-computer simulation is that, with topics such as these, the participants *choose* the activity rather than having it forced upon them, and are therefore likely to become more involved in the simulation and thus derive more from the experience.

SLICK! - a typical simulation

SLICK! will serve as a typical example to illustrate the pros and cons of this type of simulation. Users are required to prevent an oil slick from causing damage to a local beauty spot, protecting as far as possible the fishing grounds, the harbour and so on.

Information about ways of handling an oil slick, with their respective advantages and disadvantages, is provided in printed form with the program, and this gives an opportunity for a considerable amount of group-reading and discussion prior to running the program. Also provided is a printed version of the map used in the simulation, with co-ordinates and compass-points marked on it.

At the simplest level, using SLICK! involves reading details of the changing speed and direction of the wind from the screen, and then plotting the movement of the slick in the form of co-ordinates on the map. This is good practice in rapid interpretation of compass-points. At a higher level, groups can get involved in policy-decisions about handling the slick, choosing methods of combatting pollution, applying them, and then reading a report on how well they dealt with the emergency.

A possible disadvantage for students whose production of English is slow is that SLICK! is a "real-time" program — once started, the clock ticks away and if action is not taken within the specified limit (one and a half to two minutes per "turn", representing fifteen real-time minutes), "TIME UP" shows on the screen and that "turn" is lost. The discussion, which should be the major reason for using the program with a language-group, is therefore curtailed to fit into the time-limits specified for an intended audience of native English speakers.

Nevertheless, with adequate advance preparation on the part of the teacher (re-organising printed materials into a more easily-digested form, and perhaps writing a simplified "dictionary" of the more technical vocabulary), and with reading and discussion within the group to clarify strategy before running the program, a simulation like SLICK! will be a beneficial language experience. More advanced students should be able to cope without any modification of the supporting materials.

Simulations for language learners

One simple simulation written specifically for the language-learner by Chris Harrison, whose BASIC listing for the program was published in *Practical Computing* magazine (Harrison, 1982), is a prototype for the kind of subject that teachers involved in language-work should be demanding. Called TOWNPLANNING, it assigns roles to up to six players, who live, work or own property in a town in which a new factory is to be built. Each character is assigned certain vested interests in fighting for the factory to be sited in a certain part of the town (the factory owner wants to build near the railway-station, for example, but will not build at all if a bypass is built). The program draws a map of the town, and allows the participants to arrange a site for the factory and plan over- and under-passes, zebra crossings and one-way systems to cope with the increased traffic. The idea is to "promote an understanding of the language of persuasion, negotiation, argument and compromise".

There are no "right" or "wrong" answers. The participants are given roles (that is, they are told who they are, and what interest they have in the topic) but they are not given "personalities", so they remain free to argue and persuade in any way that seems appropriate to them in the given situation.

In a classroom simulation, the "Controller" (who may or may not be the class teacher) hands out new documents and information during the discussion — usually at set times, regardless of the progress of the groups. In a computer simulation, the computer uses the choices made by the participants to compute the next stage of the game, so their choices affect the outcome of the simulation (usually by making them into millionaires or paupers, or deciding whether they live or die). This can be done either in response to a request from the group (so the group exercises control over the speed of progress), or the program can throw up occasional random events (perhaps in the form of a newsflash or a telegram) to colour what happens in the verbal exchanges, or even change the direction of the "storyline". The computer does not act as an arbiter, but merely displays (in the case of TOWNPLANNING, this means marking on the map) the choices made, without attempting to interfere with the progress of the dialogue between players.

The Heinemann DUDLEY PROGRAMS for children also contain a town-planning simulation, along with some other programs which are suitable for adult language-students, such as CAR JOURNEY, which involves planning a route on a map (a "real" map of Britain, showing main routes) avoiding certain constraints and hazards.

The value of simulation in language learning

Simulations can make demands on the users in a variety of ways which are useful to language-learners, especially when used as a group activity.

There is a preparation stage, including the considerable work involved in reading the instructions and understanding what to do (before the computer is even switched on), and the further preparation of each new "frame" of text, which often involves a member of the group reading aloud the fresh information.

Then there is a discussion-stage, when each member of the group can contribute ideas about handling the situation. This exercises various types of language-output, including the language of suggestion, mediation, negotiation and possibility.

Follow-up activities (not involving the computer) might include a writing-stage (in which the group could produce a written report of the simulation or generate and record a dialogue based on it), a role-play stage, and perhaps even a de-briefing in which players recall the choices made and speculate on what might have happened if a different course of action had been chosen.

There are many "real-life" subjects for simulations and role-play which would interest and stimulate language-learners, but which have so far not been exploited. The potential for both adventure-type games and computer simulations in language classes is enormous, but they belong in the next generation of language-based programs. Fortunately, "a generation" in computing terms is currently running at about two to three years!

9. Aids to creative writing

Most of the programs mentioned so far involve the student only in typing a letter of the alphabet, a number, or possibly a word or phrase. The "reward" of the programs lies either in what the computer gives back on the screen in return for the user's input, or in the exchanges which take place among users prior to the input. Hardly any programs yet allow the student to be "creative", by conferring the freedom to put into the computer any input which is suitable to the occasion. If the student responds in an unexpected way, the program will usually refuse to understand, and will often come back with a hostile message. Few programs — possibly *no* programs — yet allow creativity, digest it, and then suggest improvements.

That is not to say that such a program cannot exist, only that, given the present state of technology and the kind of people who are currently involved in the exploratory programming of CALL, the requisites for experimentation in this field are difficult to come by.

Nevertheless, one kind of program already exists which can open up new territory in creative writing, and that is the word-processor. The kind of word-processor which is most appropriate for language or literacy students to use will be discussed in a later chapter, along with word-processor-based programs which may go part of the way towards achieving the goal of programs which accept truly flexible input. For the moment, commercially-available standard word-processors are on the market which can transform the way students write, and even encourage writing skills in those who do not willingly put pen to paper.

What the word-processor can do

A word-processing program can best be described as a series of electronic "blank sheets" which may be written on, erased and rearranged at will, before a final copy is committed to paper. Few people who have grown accustomed to the facilities of a word-processor would wish to regress to an ordinary typewriter. To quote Seymour Papert's *Mindstorms* (Papert, 1982):

> For me, writing means making a rough draft and refining it over a considerable period of time. My image of myself as a writer includes the expectation of an 'unacceptable' first draft that will develop with successful editing into presentable form . . . (but) for most children rewriting a text is so laborious that the first draft is the final copy, and the skill of rereading with a critical eye is never acquired.

The same goes for most adults — even those who have access to a typewriter. The word-processor, however, can change the writing-habits of a lifetime. After typing in one version of a text, the writer can:

☐ insert or delete at will;

☐ change the order of phrases or paragraphs, to reorganise material more logically or simply to compare the effectiveness of two versions;

☐ find out if a certain word or phrase has been overused, and alter some examples of it;

☐ re-format a whole text, or part of it, perhaps changing margins or changing from single to double spacing.

Most commercially-available word-processors offer these basic facilities, and many more besides. They can be used to edit text. They can also be used as "ideas-processors", to jot down notes which will later be expanded and refined into a full text. You can even ask some word-processing programs to check spellings and comment on anything that looks suspicious — such programs do not tell you how to spell words, but merely ask whether an unusually-spelt word is what you really meant.

Word-processing for students

Using a word-processor, any student (even a two-finger typist) can readily learn to produce clean, neat texts, and this in itself is often a new experience. After discussion with class colleagues or with a teacher, the student can then revise the text and produce an equally clean "second edition" with very little extra work beyond that involved in deciding what changes are to be made.

The "professional" appearance of work produced on a word-processor is incentive enough to many students to raise their output of written material phenomenally, and the increase is qualitative as well as quantitative.

Up to now there are comparatively few programs specifically written for English language or literacy students, and not too many which are entirely suitable for "borrowing" from other disciplines. But one program alone justifies the expense of a computer, and that is a word-processing program. For those who have physical or psychological difficulties with handwriting, and for any student who has never used a word-processor before, it provides a chance to produce clean, readable copy, possibly for the first time in their lives; for those few fortunate enough to have used a word-processor in school, it relieves them of the frustration of returning to pen, paper and rubber; for those who may be planning a career involving office-skills, it gives an insight into the practical value of modern office technology; and for everyone, it gives an opportunity to explore and play with language in a way that no other classroom activity can do.

Word-processing for teachers

Word-processing can be valuable to teachers as well as to students, and it is appropriate to discuss at this point the ways in which a word-processor can help to transform the production of teaching-materials as well as revolutionising the writing-process for students.

In my own department, the word-processor is used by most of the teaching staff. There is a "bank" of word-processed materials available to everyone: a new teacher can learn, with about ten minutes' instruction, how to "tap" existing materials and most then go on to learn how to create their own materials. A few decide, having given it a try, that word-processing is not for them, but even these teachers sometimes draw materials from the communal "bank". Many teachers have their own discs, on which they keep their personal materials.

Whether for personal or shared work, there are many advantages in storing teaching materials on discs rather than on paper:

- ☐ materials can be updated as often as necessary, without retyping the whole document;
- ☐ files can easily be printed straight on to, say, a Banda master, and class-sets run off as often as needed, without retyping every time the stencil wears out;
- ☐ copies of disc files may be made with ease — it is possible to alter a single word in a document, or add or eliminate a question, or provide two versions of worksheets, one incorporating sample answers, or make a cloze-version of a text — while still retaining the original copy; this means that teachers with very different approaches to teaching may make use of the same worksheets in widely-differing ways.

Our "bank" of materials on disc includes a selection of worksheets for use with audio-tapes, reading-texts with their associated worksheets, song-words, and (a few) structure-worksheets. The ease of copying disc-files means that teachers can share materials more than ever before, simply because there is no need to hoard single, poorly-typed, hard-won copies.

One paper-copy of each "bank" file is stored for central reference, and each is labelled (using labels designed and mass-produced on the word-processor) with all the information necessary to link up the associated disc-file, book, transparency, tape or whatever, so to a certain extent the materials are "self-indexing", which means that more people use what is available simply because they know of its existence.

The word-processor is the nearest that we yet have to an all-purpose, all-things-to-all-users program. There is no doubt that it will in future become an even more versatile tool than it is already, and that word-processing techniques will form the basis of many of the language-programs of the future.

10. "Total Package" software

In the infancy of language laboratories, EFL teachers using primitive equipment in surroundings that were far from ideal spent hours making amateur recordings. Something very much like this is happening with computer programs today, with self-taught programmers putting together programs in their spare time. Some of the teacher/programmers have had their work published, but many are content to serve the narrow audience provided by their own establishment.

The next stage in the history of the language laboratory was the intervention of large publishing houses, who foresaw a massive market in "convenience" software for EFL, and stepped in to supply that market with package-deals comprising teacher's book, student's textbook, student's workbook, taped drills (with tapescript), taped dialogues (with tapescript), and the rest. The result was in some (though by no means all) cases a more professional-sounding set of tapes, but for many students "integrated software" meant the death of any interest they might have had in the language laboratory. The language lab. became just another extension of their course-book.

Very few teachers now record their own material to use in the lab. They pick up the tape that goes with the unit they are "doing" that week, and as a result teachers as well as students find the whole business uninspiring in the extreme.

Will computers die the same death? At the moment, it is not possible to order the course-discs along with the course-tapes and the course-books. My personal hope is that it remains that way, and that the day never dawns when language-students troop into the computer-laboratory for another session of the course-book computer-drills. Computer programs must not be degraded into mere back-up for other types of classwork. The computer can provide an unique opportunity for language development work, but its potential will be thrown away if the bulk of computer software is tied to textbooks, especially if this means "one student-one machine" computer laboratories, cast firmly in the programmed-learning mould.

The way forward for computers in language work must be to develop a wide variety of individual- and group-study materials for use in a workshop or library, or in classroom group-work. Many of these programs will be "one-off" efforts, in the sense that they do not need to be linked with any other materials. Others will be "integrated" in that a set of programs may draw on the same data (via an authoring system), but the programs will still be independent of other course-materials.

The computer is, for the present anyway, student-controlled, and the student-user decides what to use, when, and for how long. It would surely be a retrograde step to take that choice out of the hands of the learner.

Chapter 2

Increasing the range of programs in CALL

Mindstorms (Papert, 1982), while it might appear on the surface to tackle the problem of getting over mathematical ideas to children, is in fact a fascinating study of how people learn — in particular, how traditional education forces its victims into unnatural learning patterns. The computer revolution, says Seymour Papert, is already breeding its own conservatism, and this is partly because we are not yet able to see what might be achieved with the new technology, and partly because society is seemingly unable to escape from a web of traditional practices which have long lost any reason for existence.

Particularly in the field of education, Papert believes, conservatism "has become a self-perpetuating *social* phenomenon", since people with imagination, creativity and drive either do not enter education in the first place, or leave it rapidly in despair. However derogatory to one's personal vanity, it must be acknowledged that there is more than a grain of truth in this point of view.

Three quotations from *Mindstorms* sum up educational computing in a nutshell:

> It is hard to think about computers of the future without projecting onto them the properties and limitations of those we think we know today. And nowhere is this more true than in imagining how computers can enter the world of education . . .
>
> The first use of the new technology is quite naturally to do in a slightly different way what had been done before without it. It took years before the designers of automobiles accepted the idea that they were cars, not "horseless carriages", and the precursors of modern motion pictures were plays acted as if before a live audience but actually in front of a camera . . .
>
> Thus, paradoxically, the most common use of the computer in education has become force-feeding indigestible material left over from the pre-computer epoch.

This goes a long way towards explaining the vast quantities of multiple-choice tests which pass for "educational computing", especially in the home-computer market. Multiple-choice still has its place in the world of education, and will continue to have a place, at least until we devise more humane methods of assessing progress. But there are better ways of using the power of a micro — ways which we have hardly begun to explore.

This chapter will not concern itself with science-fiction, only with those aspects of computing which could be within our grasp, if society is prepared to underwrite their development.

Many of the programming techniques are available to us now, using the computers we already have. Simulated dialogues, simulations, and adventure games already exist, and are becoming more sophisticated as the capacity of microcomputers to store and analyse large quantities of text increases. This chapter attempts to see these program-types, and others which are readily available, such as word-processors and databases, from the point of view of language-acquisition, and suggests ways in which they might be adapted in form and content to suit language learners. Some of the suggestions would involve considerable feats of programming, but they are not impossible. There are two factors worth bearing in mind throughout the discussion which follows:

- ☐ the development of "natural language" programs is progressing at a phenomenal rate, particularly in the world of business, and it makes sense for language-based programs to take advantage of the possibilities which are being opened up;
- ☐ there would be a spin-off from the development of "natural language" programs for language learners, in that the programs would be flexible enough to prove valuable across the curriculum.

Simulated dialogue

Simulated dialogue is a technique which could be used extensively in language-work. The earliest and most famous example is ELIZA, which was originally produced on a very large "mainframe" computer, although some microcomputer versions have been written. In ELIZA, the computer plays the part of a psychiatrist, and the user is the patient. The "psychiatrist" asks questions, and the "patient" types suitable responses. By picking up key-words from the user's replies, the computer can generate the next stage of the dialogue in a realistic way. The sinister side of this "fun program" is that some of the subjects who have tried ELIZA have been convinced that they were having a genuine and meaningful dialogue with the computer, and have felt able to communicate with the program in a way which they would not have dreamt of doing in the presence of a human being.

While ELIZA may be "fun" for most users, it can be so realistic that there would be obvious psychological risks involved in making a program on such a topic freely available to students, but there is no reason why the technique should not be used in less sensitive subject-areas.

CHATTERBOX is a dialogue-generator with an authoring-facility, so that a variety of dialogues may be constructed within the framework of the program. One of the situations created by the British Council enables the user to participate in an interview for a place at a college or university, the computer taking on the role of interviewer, and asking questions about the "candidate's" educational background, leisure interests and so on. Other possible subject areas could include job interviews, visiting your child's school to talk to the teacher, or role-plays in which the computer could play the part of a radio-interviewer talking to one of a range of "personalities" played by the user, who could assume any role from a politician to a pop-star.

This kind of program is not of lasting interest when played by a single user, but when a group of participants "takes on" the computer, much discussion can be generated about what the responses should be, and part of the fun is in finding out what the program's limitations are. Obviously, the computer cannot "know" everything there is to know about the chosen subject, so when faced with a situation which it is unable to recognise, it becomes non-committal, and uses such let-outs as "Tell me more . . ." or "Could you explain that, please?"

Dialogue-generators are not super-intelligent programs. They contain data consisting of keywords which are likely to arise in the course of a conversation on the one subject they "know" about, and a selection of standard responses to those keywords. They operate on the very simple level of picking keywords out of the user's input and making a random choice from the stock of suitable responses.

The keywords must be carefully chosen, and must contain a wide selection of vocabulary relevant to the subject of the dialogue. There must also be as wide a choice of responses as possible, to make the dialogue more realistic and to avoid monotony. Good programs will also anticipate the kind of thing that might be input in order to "trick" the system, and will provide suitable replies.

These generators know nothing about spelling, or about structure. If a keyword is mis-spelt, the computer will not usually try to guess what might have been meant, and will simply come up with one of its non-committal answers, which may be a bit disconcerting if the typist does not realise that only the spelling is at fault. However, the problem of poor spelling is less likely to arise if the dialogue is a group-effort, since one or other of the participants will probably spot errors as they are typed.

Conversely, if the keyword is spelt correctly but the remainder of the input is structurally unsound (or even gibberish), the computer will still pick out the keyword and act upon it. The structural problem is not really a problem at all, since the object is not to get students to produce structurally-perfect utterances, but to communicate. If the input really is gibberish, it is probably a deliberate act of attempted sabotage on the part of the users, and may be disregarded!

One way in which the performance of simulated dialogues might be improved for language-learners is by the addition of a spelling-check routine, which would give the program the ability to handle spelling mistakes. The only purpose of introducing a spelling-check would be to improve the flow of the program, not to issue a reprimand every time an error was made. It need not always draw attention to the error (it could simply internally substitute the correct word for the mispelled one), although in some cases (where there might be a possibility of ambiguity, for example) it might call for the user's confirmation as in a real conversation: "Sorry, I didn't get that — did you say 'school'?"

Why use simulated dialogues?

The question remains, why bother to do all this on a computer? What is wrong with the time-honoured tradition of pairing students and assigning roles to them?

Traditional role-play serves its purpose in language-work. But the purpose of computer role-play is different. It allows a small group of people to assume a role collectively, and discuss in some detail what responses might be appropriate to the situation, while the computer waits patiently and silently until required to contribute again; this does not happen in traditional role-play. It also ensures that at least half of the dialogue will be couched in good English which is more or less relevant to the situation, so that the users can see the lines on which such a conversation might develop in the real world. And it provides an opportunity for the users to see how far they can push the computer before its spurious realism breaks down.

Having seen how the computer handles the situation, students will be in a better position to enter into traditional role-play, bringing to it some of the techniques which the program has used to elicit and respond to information.

Simulation

> . . . Very few are bright, well-motivated students from the middle and upper classes. Most of them have a working-class background. Some of their fathers are unemployed, and one or two are in prison. When they leave school, most of our students get jobs in factories, or cleaning the streets, or are unemployed.
>
> Our job is to do the best we can to give them an opportunity in life. The important thing is to help them tackle their examinations . . .
>
> The problem with these classes is to find enough time to cover the basic course. There is no time for optional extras, particularly fun and games. Our job is not only to teach language, it is also to see that there are suitable conditions for learning the language.

The above quotation is not from a dedicated ESL teacher in an inner-city area. It is part of a role-card from a simulation about the effectiveness of simulation as an educational technique in language-learning. The complete simulation, called *We're not going to use simulations*, appears in *Simulations in Language Teaching* (Jones, K., 1982), in which "Teacher Kay" (whose role-card is reprinted above) plays devil's advocate.

In the opinion of Ken Jones, the object of a simulation is to provide a setting in which students can function effectively using the language they know. To counter "Teacher Kay's" arguments, simulations must be demonstrably more than just "fun and games" — they must be relevant to the students' needs; they must help the students to function effectively in situations which are "part of the basic course"; and they must generate "suitable conditions for learning the language" which are often *not* found in the average "teacher-in-front-of-the-class" environment. To quote another author, Leo Jones, writing in *Eight Simulations* (Jones, L., 1983):

> In a simulation, participants find themselves using all the English they know to achieve their aims — just one step away from what will happen to them in real life.

"Teacher Kay's" argument, all too frequent throughout the teaching profession, about the need to prepare students for examinations is forestalled by Ken Jones' observations:

> Since the language classroom is becoming more and more a location for student participation, talk and decision making, it seems only a matter of time before written language examinations include non-interactive simulations.

"Non-interactive" simulations

The kind of "non-interactive" simulations he has in mind are variations on the "In-Tray" theme, which is a one-person exercise. The participant becomes an office-worker, a bureaucrat, a local councillor or whatever, and has to handle an appropriate "in-tray" for the duration of the simulation. This may involve writing letters, reports and memos, sending telegrams, and so on. Without benefit of a computer simulation, this may be done with the teacher acting as Controller, handing documents to the participant, and the participant producing responses in handwriting, or, better still, on a word-processor.

In a computer-aided version, the machine would act as Controller, providing on demand printed documents which would require interpretation and action by the participant. A built-in word-processor could store the responses and offer varying levels of assistance, and the participant's responses would finally be printed out in appropriate formats (memos on "memo-forms", letters on "headed" paper, newspaper articles formatted into columns, or whatever is required). A more sophisticated program might extend the simulation to include "real" computer-based activities, such as a database from which details must be extracted in order to complete one of the writing activities.

Group simulations

The bulk of simulations are not solo exercises but involve co-operation of various kinds between participants. Some simulations of the "solo" kind — such as the "You are the King of Ruritania . . ." type — benefit enormously from being tackled by small groups rather than individuals, but even in this form they may be less productive in language terms than those in which each participant adopts an individual role (as in TOWNPLANNING) or those which rely on the concensus of participants (for example, a group with a common aim, whether it be making a fortune or escaping from a hostile environment).

Whatever the type of simulation, the computer adopts the role of Controller, with the function of presenting new materials at appropriate times, and "handing out" documents, either on screen or via the printer. This may add to the realism of a simulation, if carefully handled, the screen being used for "newsflashes", the printer for "telex" messages, and a linked cassette-recorder for "radio bulletins", all under the control of the computer program. The potential danger in this is that the teacher who normally adopts the role of Controller also serves a dual purpose as Assessor. While the computer makes an excellent Controller, it is not in a position to assess performance or to organise a debriefing session. These tasks remain with the teacher, who must not feel that all responsibility can be abdicated while the computer takes over the teaching role.

Borrowing simulations

Simulations, as has been indicated already, are available in large numbers, both as "games" and for educational purposes. Borrowing simulations from other disciplines can be useful in that, while investigating a subject which is of interest in itself, the user is exposed to the language used by English people for "real" communication. This is a welcome change from many classroom activities which are generated solely for the purpose of illustrating a particular function or structure of language.

However, many of today's computer simulations are limited to the extent that they rely heavily on numbers, and therefore work well only with subjects which lend themselves to numerical calculation — for example, if you adopt the role of a farmer and decide to plant 25 hectares of wheat, your crop can be calculated according to a formula and then moderated, again by a numerical formula, to allow for the occurrence of certain random events, such as an unexpected frost or particularly good harvesting conditions.

Classroom simulations involving situations which are essentially socially based (such as editing a newspaper or deciding environmental policy) rather than numerically based do not lend themselves so readily to computerisation, and have therefore been largely ignored, but that does not mean that they are impossible to create, and there is certainly no reason why simulations should not be geared to the specific needs of language students — and, indeed, why these should not then be borrowed for use in other areas of education!

Teachers who wish to use computer simulations must be aware of the possible risks, however. One of the risks is that of over-simplification: students might be inclined to believe that a computer simulation is a model of real life, when in fact the programmer has chosen to present a simplified version, either to avoid programming difficulties, or because it seems easier to understand. The second and much more serious risk is that students will believe that simulations are objective, when in fact they are only a reflection of the programmer's (or programming team's) subjectivity. Take, for example, the common type of simulation which involves "being the Prime Minister" and "running the country": economists in real life will argue endlessly over the effects of raising taxation or increasing public spending, but a simple microcomputer simulation must decide on the "effects" within the program on the strength of a simple mathematical calculation, and the nature of that calculation will depend on the subjective political opinion of the program planner. Daniel Chandler (Chandler, 1984) comments that:

> . . . the underlying model may be deceptive or misleading, or simply inaccurate. Even more importantly, it may be hidden from users, who ought to question its validity, but may neither think of doing so nor know how.
>
> . . . The use of computer simulations makes it essential for users to be conscious that the models they use are only personal and selective representations of aspects of the world.

The teacher therefore has a responsibility to the students to ensure that they understand how far they may depend on the accuracy and validity of the computer's reaction when they are using a serious simulation.

A simulation experiment

A simulation under development at the City of London Polytechnic shows what can be achieved using standard programming techniques allied with a great deal of imagination. This program requires students to take on responsibility for a number of visitors who are travelling to Britain from various parts of the world in order to attend a conference. Some visitors are travelling alone, some are bringing husbands or wives; travelling arrangements must be made on suitable flights, cars must be arranged to pick up visitors at the right time from the right airport, hotel rooms booked to suit the budget and special requirements of each visitor. From time to time, random events cause a change of plan — one visitor may decide to bring an unexpected travelling companion, another may be recalled home suddenly, or a plane may be diverted or delayed.

All this is achieved through a combination of flight timetables, hotel booking schedules, telegrams and so on, with a minimum of typing from the "organiser" at the keyboard. The language work takes place *within* the group of users, and the amount of "language" needed to communicate with the machine is negligible. On the other hand, the computer plays a vital part in the simulation by keeping tabs on decisions made by the students, displaying quickly and accurately whatever information is needed at a given moment, and occasionally throwing a random spanner in the works to maintain interest.

This simulation works well because, although the subject is not numeric, it is tabular, so that the main task of the program is simply to maintain, update and display little databases of information about each of the "visitors" as the students make their decisions.

Simulation language

The travel-arrangements simulation is unusual in that its language-content is high, and of a high quality, but some popular commercial simulations have very little inherent language-content. The screen presentation is in the form of charts and figures, labelled with abbreviations, and the instructions are minimal, often in an elliptical form of computer-English.

As long as the group understands its task, these need not be insurmountable problems, since the object is for the members of the group to communicate with each other, not with the machine; the computer is merely a background to the group's activities, and a passive receiver of its instructions.

Nevertheless the language produced by the computer may cause soul-searching amongst some teachers. Where simulations involve descriptive prose, the style and subject-matter, not to mention the spelling, can leave a lot to be desired. The number-based simulations tend to cut down description and instruction to a bare minimum, and what little there is, is often couched in the Computerese language, which is only loosely related to English.

In the opinion of some teachers, this is an unacceptable burden to place on those who are already struggling with an unfamiliar language. Yet we often introduce our students to different styles and registers of English; we ask them to understand, and even produce, notes and summaries, telegrams, postcards and advertisements. In one respect, computer-English is just another writing-style, and one which will become increasingly important in the lives of all of us in years to come, whether we like it or not.

"Natural language" simulations?

There is no reason, however, why simulations intended specifically for language-work should not present the users with natural English, such as we might normally use in the situation which is being simulated. The instructions, too, need to be as full and as clear as possible, to suit the language capabilities of the target users. Only a short time ago — a matter of a couple of years or so — a microcomputer-memory of 16K was counted as generous, but nowadays 56K or 64K is the norm, and the next generation of educational machines will have a standard 256K or more. Without entering into details of precisely what "256K" represents, suffice it to say that, with memory no longer at a premium and disc-storage rapidly becoming cheaper and more capacious, it is no longer necessary to economise on program-size and so programs which draw on large amounts of text become a viable proposition.

A language-based simulation should therefore present not only the text but also the prompts in natural language, using upper/lower case and correct punctuation, as appropriate. Even in a very simple situation, a variety of different ways of asking for information could be incorporated, selected at random by the computer in the course of the simulation, so that users are subjected to a range of stimuli.

So much for the language produced by the machine. The language that the *users* are required to contribute to the simulation is rather more problematical. Most simulations ask for a very limited input, often no more than a number or a letter typed from the keyboard in response to a prompt such as "NUMBER" or "WHICH ONE", or even "?". This makes life very easy for the programmer, but it does limit simulations to subjects which are numeric in nature (like SLICK!) or tabular (like the travel simulation described above). A simulation which demands natural language from its users has yet to be written, but when it is, it will open up vast new areas of subject matter for language-programmers to explore.

Subjects for simulation

Nearly all the readily-available simulations require the users to participate in an activity which they could not expect to experience in real life, like voyaging round the world in a sailing-ship or fighting the Wars of the Roses. For native speakers, the value of the simulation is its subject-matter; the last thing they have in mind is exercising their knowledge of English, though for some users that may be a useful by-product. But while many of these subjects may be of general interest to language-learners, they contribute little to their ability to cope with everyday life in the twentieth century.

Ken Jones (Jones, K., 1982) suggests that the subject-matter of a simulation is unimportant so long as the motiviation is there, and motivation comes from "acceptance of the duties and the desire to function effectively." It is necessary, though, to consider subject-relevance along with motivation, since in many learning situations the two factors are interdependent, especially where time is limited. A simulation which requires adult students to put themselves in the position of scientists, politicians or poverty-stricken inhabitants of the Third World in order to plan future expenditure on space exploration is far more likely to "work" than one which requires participants to play-act as explorers in space. The "relevance" of a subject should not be measured in terms of "how likely is it that this student will ever be a politician or a space explorer?" but on the basis of "how far will this topic contribute to students' understanding of the world in which they live?" By this measuring-rod, a wide variety of topics of social, cultural and political importance will come under the heading of "suitable subject matter". This applies to almost any subject area, not just language-learning.

But the language-learner also has a special need for simulations of the more prosaic situations found in real life, in which the understanding of language will be called upon. This need can be met by applying ordinary simulation techniques to suitable subjects of the "What would you do if..?" variety, or by allowing an extended input of the type found in an adventure game.

The participants may, for example, have to look (collectively) for a job, or find a room to live in. The real language-work would take place within the group, as they discussed what course of action would be best in the given situation.

"Looking for a room" – a "real life" simulation idea

In a "looking for a room" simulation, the "given" information would include the searcher's income, place of work, and a map of the area. The program might offer a newspaper, an agency, or a post-office window as starting points, and the initial discussion would be on the relative merits of the three modes of advertising, and which would be most likely to be a fruitful source of suitable accommodation. If they instruct the program to "look in the newspaper", a "page" of advertisements would be offered, using the usual conventions of abbreviation, giving details of rooms to let, requiring a decision about whether to phone one of the numbers given, or visit one of the addresses to make further enquiries about the rent, the facilities offered and so on. If they choose to visit an address, they would need to consult the map in order to find out the best way of getting there, and whether there was suitable transport to and from work. A groundplan of the room and a description of facilities, together with a local map, would form the basis of a discussion of the suitability of the accommodation. The possibilities are endless.

In this kind of simulation, the computer does not generate an "original" response to the user's instructions, but simply searches its "bank" of possible storylines for the one which seems most appropriate to the developing situation, in much the same way used in a simulated dialogue.

Going shopping" – a "natural-language" simulation idea

A very different kind of simulation could both generate language suitable to the given situation, and also require the player to use the precise language that would be required if the situation were real, but this would involve the computer as a *participant* in the simulation, not as the impartial Controller.

Suitable subjects might include buying a railway ticket, shopping, booking into a hotel, finding your way around a strange town and so on. The situations need not be complex, although the programming would be. "Shopping" might involve something on these lines:

☐ the computer offers the user a random amound of "money" to spend, and a shopping-list, which could be printed out for reference during the program;

☐ the object is to "buy" as many of the things on the shopping list as is possible within the budget allowed, by asking for them in an appropriate manner;

☐ the computer might respond by asking for the size, colour or quantity required, by saying that the item is out of stock and asking the player to suggest an alternative, by offering different qualities of the same product at wildly varying prices, or by trying to tempt the customer to buy something different or more expensive;

☐ the "customer" questions the computer to get details of what is available, and decides what to buy, based on a "budget", which the computer updates as purchases are made.

Keyword-matching

What is involved in such a programming project? There are two possible approaches. One would treat the simulation as a kind of enhanced "simulated dialogue", the difference being that the simulation requires a task to be completed (in this case, the shopping done within the given budget), whereas the dialogue is open-ended. The computer would generate its own part of the transaction in much the same way as in a simulated dialogue, by looking for keywords in the user's input, but in addition the machine would have to elicit certain information about each product before it could proceed with a "sale" (it would not be satisfied with selling a pair of shoes, but would ask the "customer" to specify the size, colour and style).

The user's input could also be handled on a keyword basis, so that it would not be necessary for the input to be "correct" as long as it was intelligible to the program. This would mean however, that to some extent the user could get away with very terse responses, often of a single word, although the encouragement offered by the computer's own production of realistic English should be enough to stimulate the user into responding in kind.

If the keyword-matching method were adopted, the program would still have to be sophisticated enough to generate suitable language to cope with the shop-assistant's side of the situation, and to recognise valid keywords in the answers to the question it generated. It would need to check the spelling of the keyboard-input, and either overtly offer to correct the spelling, or silently substitute the correct version for the wrong one. It would need to be taught even the simple things which to the human brain are obvious, such as the fact that "six" and "6" are to all intents and purposes the same thing.

Language-parsing

Keyword-matching is the "simple" way of handling a language-based simulation. The other method is to write a language-parser. Before any attempt at "coding" (putting the ideas into a machine-readable language), the language of shopping would have to be analysed and the results tabulated. A range of products must be chosen at an early stage, since these might affect the choice of the vocabulary and structures used by the program (it would be simpler to ask the user to shop for a range of groceries or a range of clothing, rather than for a mixture of groceries and clothing).

Even limited in this way, the computer could not be fed with every possible utterance that might occur, so the permissible structures must be broken down into manageable chunks which could combine with the permissible vocabulary to form reasonable shopping-situation utterances. The parser's function would be to check the user's input against its own "knowledge" of what people say when they are shopping, and either suggest improvements, or interpret and act upon any input which is perceived as "correct".

Unlike an adventure game, where part of the fun is to find out which words the program recognises, the simulation would include a printed dictionary of its complete vocabulary, and the emphasis would be on finding acceptable ways of using the vocabulary provided. A very simple parser for this situation might include only such basics as "Have you got..?" and "I want the large size, please", and could require the user to be specific about the purchases (it would be possible to buy a box of matches, for example, but not a box of carrots); a more substantial parser would allow much more complex utterances, like "What's the difference in price between the standard and the de luxe models?"

Both methods are possible using today's equipment and programming skills, but the second method is a great deal more complex a programming task than the first, and has hardly been attempted so far in language-work, although examples exist in other areas, notably in business. Computers with a large enough memory for this kind of program are rarely found in education today, but within a comparatively short time large memory-sizes will be the norm rather than the exception.

Summary

Two kinds of simulation have been described. The first kind is a "What if..?" story, and most simulations fall into this category. Their value to the student is in promoting group-discussions about the best course of action to take. Unfortunately, many subjects for these simulations are of little value in "the real world", and there is a need for a new range of topics which will be closer to the everyday experience of the target users. The City of London Polytechnic travel simulation is a model of what can be done with this kind of subject.

In the second kind of simulation, the computer will generate appropriate responses based on the extended input of the users. The value to the student in this case lies not in the processes leading to decision-making, but in group-discussion of the precise nature of the language needed to produce the desired result. Both types have a part to play in language-work, and we have hardly begun to exploit the exciting possibilities which can be opened up for language-learners by the use of language-based simulations.

Adventure games

Many teachers and more advanced students will be satisfied with the existing adventure games, and will extract a lot from them in terms of round-the-computer communication. But even those who are satisfied with the games the way they are may still welcome a future generation of adventure-type games, in which length, subject-matter and restrictions on student-input can be tailored individually to suit language learners. Certainly, students at an elementary level could then be involved in adventuring for the first time.

Length

"Adventure generators" are now available for some makes of computer, and these allow home-made adventures to be constructed by non-programmers using a personal choice of storyline and vocabulary. Length may be an important criterion if students can not have sufficient access to machines to ensure that they will be able to "book" another session. If the vast size of commercial adventures is considered to be a problem, this is easily overcome when using a generator by making the story simple enough to complete in, say, one or two hours of playing time.

If the students can have more or less unrestricted access to machines, adventures can be more substantial. Adventure generators, as well as ready-made adventures for the domestic market are provided with a facility to save the current state-of-play so that the players can return for another session on a future occasion, but since these programs are intended for individual home-users, no provision is made for large numbers of games to be in progress at any one time. In practical terms, this means that every group needs its own personal disc or cassette on which to save its "state of play".

An adventure purpose-written for the educational market should ideally provide a save-facility such as the one in the ILECC adventure, THE DARK CASTLE, which allows fifty groups to have games in progress at any one time, using only a single disc. This facility can easily be provided on disc-based machines; in the case of cassette-based machines, the only practical answer is to give each group its own cassette on which to save the game.

Subject matter

This second area of difficulty is one which cannot easily be solved unless teachers and programmers working together can come up with a new range of subjects specifically intended for language-learners. The standard adventure is either gnomes, magic and diamonds, or Klingons, deadly rays and spaceships. This sort of topic may be fun for advanced students, and will certainly generate discussion within groups of users. Whether these subjects are of lasting value is a different matter. Particularly in ESL groups, where acquiring a working knowledge of the language is a matter of urgency and necessity, learning materials need to be oriented towards more practical situations, and this may be achieved by the use of an adventure generator.

Input

The third problem of adventures is their (ab)use of language. We do not start an English language course for beginners by teaching them to write telegrams, because understanding the language of telegrams requires a prior knowledge of the underlying structures of the spoken language. For the same reason, introducing the language of conventional adventure games to students at an elementary level means teaching them to run before they can walk. Nevertheless, adventures are an interesting and stimulating language activity, and it is a pity to deprive elementary language learners of the adventure experience.

The following description of an experiment with an adventure generator will illustrate the language difficulties encountered in adventure games generally.

THE QUILL is an adventure-generating program for the Spectrum computer. With patience, imagination and logic, users can generate original adventure games to their own specification. This kind of generator provides a solution to two of the potential drawbacks mentioned above: you can generate short, simple adventures as well as long, complex ones, and you can tailor the subject-matter to suit your students' needs and interests. You can even make limited progress with the language-problem, although this cannot be solved completely.

The theme chosen for the experiment was "getting up in the morning". The player's task is simply to get up, get ready to go out, have breakfast and leave the house, remembering to take the front-door key. This is not much of an "adventure", but much more practical for an elementary learner. As in any adventure, progress depends on fulfilling the tasks set in the program, in the correct order (found by trial and error) and using acceptable language (also found by experimentation). In the "getting up" program, for example, the key of the bathroom cabinet (kept in a box on the bedside table) has to be taken into the bathroom before the player can get a toothbrush.

Like most other adventures, those generated with THE QUILL look for a two-word command from the player. If more than two words are found, the command is scanned to find the first two words that the generator recognises (usually an imperative and an object, or an imperative and a direction); it doesn't trouble to look any further. Thus, "take the key" is an acceptable command, because the program doesn't know the word "the" and simply ignores it. On the debit side, "take key" is also quite acceptable to the program, and nothing you can do will persuade it that the definite article is necessary. In the same way, "put hat" is adventure-language for "put the hat on", but what the program really prefers is "wear hat".

Most English people would not say "take the key" when what they really mean is "pick the key up", and this makes another problem. The word "take" is built into THE QUILL and although you can specify synonyms, it will continue to respond to "take" if that is what the player types. Furthermore, if the required input is "pick the key up", the program, which is only scanning the line for two key words, will respond to "pick" and "key", and will not care whether "the" and "up" are present or not. "Pick key" is even less acceptable than "take key"!

"Up" is in itself a problem, because "up" is built into the program as a direction (along with "down", and the four points of the compass). In the command "pick the key up", this is irrelevant because the computer has already found two key-words ("pick" and "key") by the time it arrives at the word "up", which is discarded without examination. However, if the command is "get up", the program looks for an object following the imperative, and instead it finds a direction. Its immediate response is "I can't do that!". Similarly, although "pick the key up" will fool the program, "pick up the key" will not.

It is possible to overcome this by eliminating "up" as a direction, and fooling the computer into thinking that "up" is something that you get, in the same way that "key" is something that you take. Now the game has to be conducted all on one level, since one of the up/down directions is missing. So the player in the "getting up" game must perforce live in a ground-floor flat!

So, while it is possible to use a generator like THE QUILL to make an adventure more suited to the language-needs of the elementary users, the result must of necessity be a compromise between those needs and the limitations of the program.

The way forward

Recognising the difficulties involved, programmers funded by the Microcomputers in Education Programme (MEP) at Chelsea College have developed an ADVENTURE GENERATOR for the education market. The MEP version does not contain any built-in, immutable vocabulary, so the adventure-maker is free to choose which words will be recognised, and what they will mean. Furthermore, the program recognises up to four keywords in the player's line of input, not merely two, which allows for "put the hat on" not only to be a valid command, but also the *only* acceptable instruction with that meaning, if that is what the writer demands. (THE QUILL forces you to accept "wear hat", whether you like it or not).

The MEP version will not only solve the most acute of the language problems, but also widen the scope of the adventure-game in the classroom. The intention is that it should be used by children, not just to *play* adventures but *create* them, too. It should therefore also be possible for adult language-learners to create and use their own adventure games as a long-term group activity, especially as the instructions are written simply and clearly.

The adventure-generator would then also become a communication-generator, as students communally plan a storyline and put it into effect. The bulk of the preparatory work would in fact be done without involving the computer at all, the keying-in of the data being only a relatively minor part.

Not everybody wants to go into adventure-game-creation in a big way.

Even though the use of a generator involves no programming expertise, it is nevertheless a job which calls for much preparation combined with logic and clarity of thought. It would be useful to have in addition a selection of off-the-peg adventures tailored to the needs of language-learners (especially at the more elementary levels), comparatively short, and using appropriate situations and language. These would certainly be marketable, not only within the rather restricted field of adult language-learners, but also in mainstream education, both primary and secondary.

Word-processing

There is no doubt that word-processing will outstrip all other uses of the micro in the language classroom, as the number of micros per group grows. At present, word-processing is the only way of using a computer which allows genuine freedom to the student. There is no comparison between the "freedom" of pencil and paper and the freedom of the word-processor. Apocryphal stories abound of students, adults and children alike, who never willingly wrote a word until let loose on a word-processor. Nor, apparently, does word-processing stunt development in handwriting — in some cases, the spur of seeing their personal work on the word-processor is all the incentive needed to persuade students to improve their proficiency with a pen.

There is equally no doubt, however, that the present generation of word-processors is unsatisfactory for students whose grasp of English is less than perfect — and also for many native English speakers who have learning difficulties, and for children in primary and remedial classes, too.

The language-processor

Most of the word-processors on the market are geared to the requirements of business, and while these may be suitable, even necessary, for some ESL linked-skills work, some students will find them complicated and difficult to cope with. Even word-processors written for educational use can be unnecessarily complicated.

The danger is that the students one would most wish to attract towards word-processing are the very ones most likely to be put off by the difficulties of operating the word-processing program. With a limited amount of time available for language-acquisition, and an even more limited time actually on the computer, students will not want to spend too long learning the basic operation of a word-processor. Even those who ultimately wish to follow office-skills courses would welcome an easy way into the subject, and all types of student from primary children upwards could benefit from a word-processor which offered basic facilities for handling language, and was transparent in use, requiring no complicated operating techniques.

A word-processor for language purposes need not offer the full range of facilities necessary to its business counterpart, nor even the comprehensive options of some word-processors designed for the general educational market. The average student will initially want to use only a limited range of facilities, and can transfer to a more complex program at a later stage when the basic version has been outgrown.

The features needed on even the most basic of word-processors are:

☐ word-wrap — the ability to type continuously, without worrying about line-endings, which are organised by the program;

☐ full-screen editing — the ability to move at will over the screen, making alterations or moving sections of text;

☐ comprehensive editing commands for insertion and deletion of text, moving blocks of text from one place to another, and using underlining and bold-face;

☐ reformatting of edited text;

☐ ability to change letters or whole words from upper to lower case, and vice-versa;

☐ simplicity of editing, using logical, memorable commands and avoiding complicated codes;

☐ a screen-display which reflects as closely as possible what will be seen on paper, possibly using colour-coding to represent features (such as expanded type) which are difficult to reproduce on screen;

A language-processor, while doing without some of the advanced facilities of standard word-processors, could made good use of some extra features:

☐ a larger character-set, providing a 20-column screen in addition to the normal 40- and 80-column screen options, so that the program will adapt itself to the needs of students of all ages and abilities. The extra-large character-set should be a sensitively enlarged version of the normal character-set, not a chunky, teletext-type design;

☐ the option of printing a text either exactly as it appears on screen, or to match the usual 80-column paper-width;

☐ a pre-printed viewing option, so that text entered in 20- or 40-column mode can be seen as it would appear if printed across an A4 sheet.

Simple editing

The editing commands of most word-processors are fairly complex, involving use of various combinations of control keys and alphabetic keys. To move a block of text from one place to another using WORDSTAR, for example, it is necessary to place a marker at the beginning of the block (the control key held down while the 'K' and 'B' keys are pressed), place another marker at the end of the block (the control key held down while the 'K' key is pressed twice), and then execute the move by pressing 'K' and then 'V', simultaneously holding down the control key.

To simplify editing in a language-processor, the ideal method would be to supply an extra piece of hardware with the software. A "concept keyboard" plugged in to the computer and placed alongside the standard keyboard would solve the problem neatly. A concept keyboard consists of a touch-sensitive pad with an overlay indicating which part to touch to get which effect. The program can "read" which part of the pad is being pressed, and take appropriate action. A concept keyboard for a language-processor might look something like this:

```
|-----------|-----------|-----------|-----------|-----------|-----------|
|  START    |  20       |  40       |  80       |   LEFT    |   RIGHT   |
|  WORK     |  COLUMNS  |  COLUMNS  |  COLUMNS  |   MARGIN  |   MARGIN  |
|-----------|-----------|-----------|-----------|-----------|-----------|
|  LINE     |  REMOVE   |   LINE    |   GO TO   |  REMOVE   |   CASE    |
|  SPACE    |  WORD     |   UP      |   START   |  LINE     |   CHANGE  |
|-----------|-----------|-----------|-----------|-----------|-----------|
|  WORD     |           |        CURSOR         |           |   WORD    |
|  LEFT     |  CURSOR   |          UP           |  CURSOR   |   RIGHT   |
|-----------|  LEFT     |-----------------------|  RIGHT    |-----------|
|  START    |           |        CURSOR         |           |   LARGE   |
|  SEARCH   |           |         DOWN          |           |   PRINT   |
|-----------|-----------|-----------|-----------|-----------|-----------|
| BLOCK     | BLOCK     |   LINE    |   GO TO   |  REFORM   |  UNDER-   |
| START     | END       |   DOWN    |    END    | PARAGRAPH |   LINE    |
|-----------|-----------|-----------|-----------|-----------|-----------|
| MOVE      | COPY      | SAVE      |   PRE-    |  PRINT    |           |
| BLOCK     | BLOCK     | WORK      |   VIEW    |  WORK     |  EXIT     |
|-----------|-----------|-----------|-----------|-----------|-----------|
```

A pad twelve centimetres (less than five inches) square could accommodate all these functions, providing an area for each function equivalent to at least twice the area of a typewriter-key. The commands are arranged logically, with the block of large "keys" in the central position representing the often-used cursor movements, and other common left/right and up/down commands placed appropriately. The remaining functions are also organised as logically as possible, from "START WORK" in the top left corner, through a sequence of formatting choices, down to saving a text, printing it and exiting from the system in the bottom right corner.

If the language-processor provided an option of two modes, "standard keyboard control" or "concept keyboard control", the concept keyboard could be supplied as an optional extra. The "standard keyboard control" mode would simply involve pressing the control key followed by one other key, the keys being chosen as far as possible as mnemonics of the required function (CONTROL + 2 = 20-column screen, CONTROL + L = word left, and so on). Many keyboards now provide a pad of arrow-keys for cursor movement, and these could be used unaltered in "standard keyboard control" mode. A really versatile program would accept either of the two modes at any point during the typing of a text.

Two other essentials for a language-processor are:

☐ plain English on-screen messages instead of jargon — for example, if the user chooses the RIGHT MARGIN option, the message might say

```
The right margin is now 35.
Type a new number between 5 and 80:
```

- plain English documentation, so that both teachers and students can understand it. It has been said that the biggest single impediment to mastery of a computer is its instruction manual. Anyone who has attempted to use the WORDSTAR manual must agree. Yet the use of a simple, well-designed language-processor should be so transparent that documentation is almost superfluous, and whatever documentation is necessary should be well-organised, logical and indexed in detail.

Educational word-processing packages

In the absence of a purpose-designed language-processor, what alternatives are available?

BBC Micro users are fortunate in having EDWORD, an MEP production, written by teachers for children. The BBC machine has a bank of function-keys in addition to its standard keyboard, and these are brought into play to provide (fairly) simple command-sequences, although the more unusual commands still require a function-key *and* another key to be pressed, as there are not enough function-keys to go round. A special "formatting mode" (called up by a two-key sequence) presents the user with an excellent on-screen display of current formatting choices — page-length, margins, tabs and line-spacing — and allows changes to be made to any or all of them. The formatting commands sometimes reflect in the screen display (margin settings, for example), but sometimes do not (line-spacing choices only show up in the printed version). EDWORD probably comes nearest of all to providing the facilities of the language-processor outlined above.

BANK STREET WRITER, an American production with versions for Apple, Atari, Commodore 64 and IBM PC micros, has three operating modes — Write, Edit and Transfer — and switching from one to the other is done via the ESCape key. This means, for example, that making an alteration to existing text involves leaving Write mode, entering Edit mode, using the cursor keys to move to the bit that needs altering, and re-entering Write mode (via Transfer mode) to make the correction. And then, presumably, doing the reverse to get back to where you were before you noticed your error. Nevertheless, written with children in mind, BANK STREET WRITER is more suitable for language applications than many other word-processing packages.

TXED, for the 380Z and 480Z, has a number of drawbacks. It takes a long time to master, time which most language students cannot afford; the appearance of the screen bears little relation to what will appear on paper; the commands are complex and unmemorable. Worthy though it may be for some applications, TXED is not really a contender as far as language students are concerned. However, WRITE, produced by Oxfordshire MEP for the 380Z and 480Z, is so simple to use that a novice sitting at the keyboard for the first time could, with a "HELP sheet" and a minimal amount of assistance, produce a creditable piece of work in a very short time.

If forced into using an unsuitable word-processing package — and that would include many very good word-processors, like WORDSTAR, which are simply not built for this kind of application — the best solution is for a teacher who is familiar with the package to prepare a brief "alternative manual", describing clearly how to start up the program, open a file, use half a dozen of the most important commands, and finally save and print a text. The advantages of making a word-processor available to language students are so great that even an unsuitable word-processor is better than no word-processor at all!

Programs based on word-processing techniques

In addition to the use of a word-processor for its original purpose of composing text, some word-processing techniques can be usefully employed to illustrate the behaviour of language. Special programs could be written to exploit these techniques, but, failing that, a standard word-processor might be used to achieve similar, if less sophisticated, results.

Letter-writing skills

A program written for learners of German consists of a miniature word-processor-like routine which "knows" how to write letters in German. By questioning the user, the program elicits details about who is writing the letter, and to whom, and various other scraps of information such as might be found in a letter sent by a penfriend. Anthony Crispin (Crispin, 1982) explains:

> At the end of this interrogation, the micro can provide the complete letter in good German, regardless of the accuracy of what was typed in (within obvious limits of intelligibility). The letter is absolutely individual to the pupil concerned . . . Depending on the teacher's intentions, greater or lesser degrees of help/remediation can be offered as the interrogation reveals comprehension or other difficulties. The final letter can be copied, or worked through several times by cloze procedure, with diminishing help being given by the micro as the pupil's answers indicate that he/she is mastering the material. If there is a printer available, the pupil can finally be given an accurate copy of the individual letter.

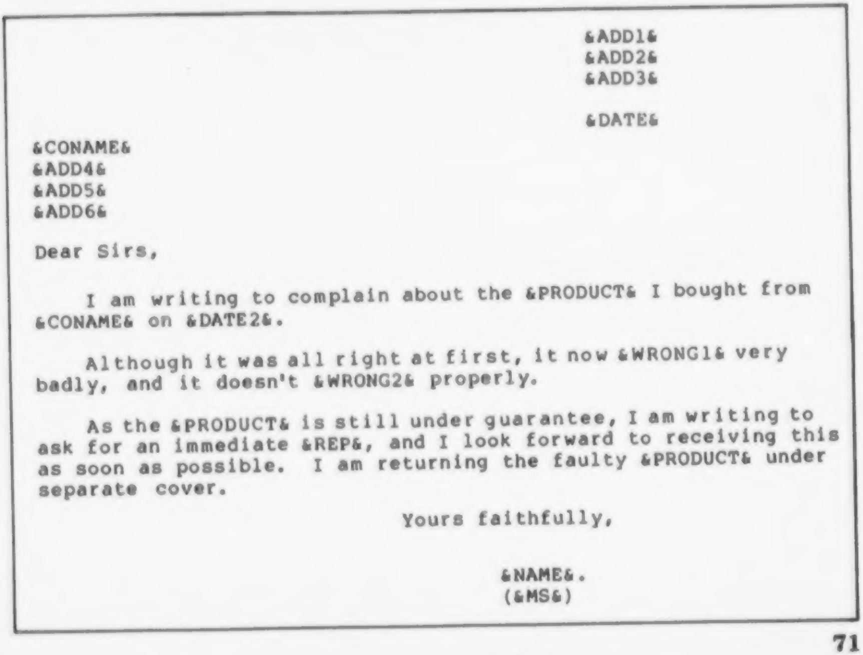
This basic idea re-written for English learners would be very welcome, even in its present pen-friend format. But imagine how useful it could be if extended to cover other types of recipient, ranging from the bank manager to the social security officer.

The word-processor function on which the Crispin letter-writing program is based is the "mailmerge" facility which is available with some word-processors. "Mailmerge" allows the user to specify what is to be included in the body of a letter, marking places where "personalised" information is to be printed. Taking a file of names, addresses and any other relevant information, the computer can then churn out "personal" letters to any number of recipients, without further intervention from the user.

Anthony Crispin's program adds a great deal to this technique, calling for information directly from the user, checking the information prior to printout, incorporating various levels of assistance, and also providing (in computer jargon) a "user-friendly front-end" — that is, a style and format which is easy for the user to understand and respond to.

However, an ordinary word-processor with a "mailmerge"-type facility can be used to simulate the Crispin program at a very simple level. It is quite easy to create the body of a letter to use as the framework of the finished product, and then build a second unit which will "interrogate" the student to obtain the required information to complete the letter. The student's input will not be checked for accuracy, the appearance and use of the program will be a trifle clumsy, and the finished product will be printed directly on paper, not initially on the screen. Nevertheless, the result would provide the student with guidelines to work on when constructing further letters, whether on the word-processor or on paper.

The following is a sample of the kind of thing which is possible using this technique. The teacher first prepares a framework, containing suitably-chosen names for the variable parts of the letter which will be supplied by the student. It will look like this:



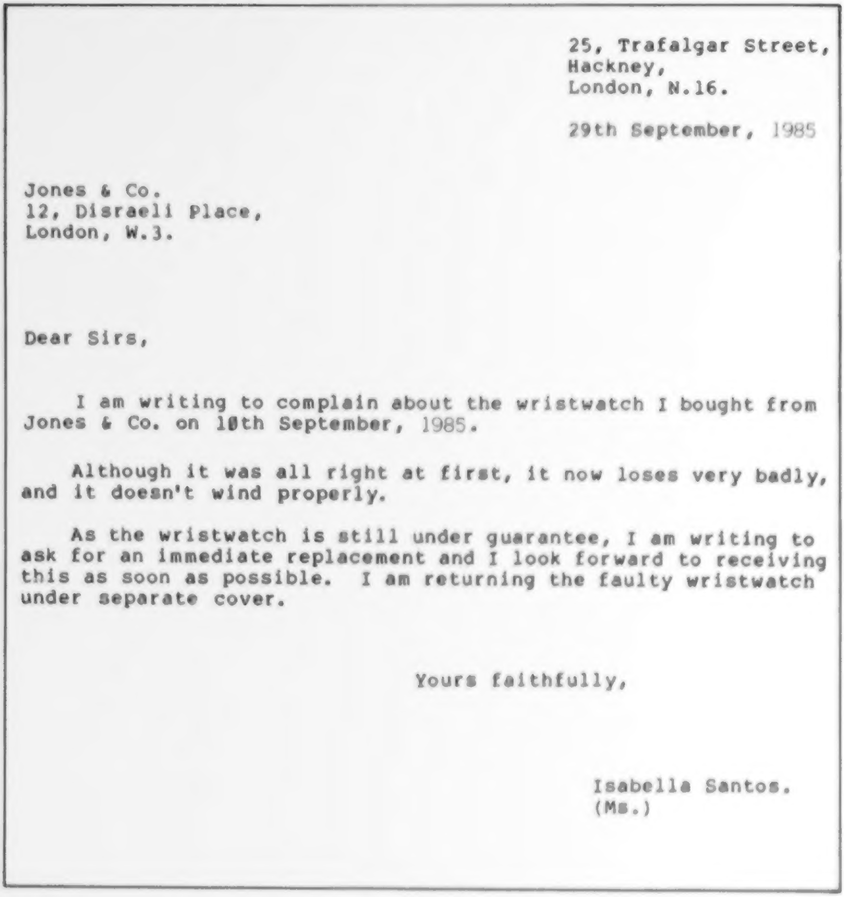
```
                                                  &ADD1&
                                                  &ADD2&
                                                  &ADD3&

                                                  &DATE&

&CONAME&
&ADD4&
&ADD5&
&ADD6&

Dear Sirs,

    I am writing to complain about the &PRODUCT& I bought from
&CONAME& on &DATE2&.

    Although it was all right at first, it now &WRONG1& very
badly, and it doesn't &WRONG2& properly.

    As the &PRODUCT& is still under guarantee, I am writing to
ask for an immediate &REP&, and I look forward to receiving this
as soon as possible.  I am returning the faulty &PRODUCT& under
separate cover.
                                    Yours faithfully,


                                          &NAME&.
                                          (&MS&)
```

This framework need not be seen by the student. The next stage is to prepare an "interrogation file", which gradually gathers together the information needed to complete the framework. The student sees only the parts between the double quotes, and is presented with one question at a time:

```
..LETPRINT.TXT - prints a letter of complaint
.AV "What is your name? ",NAME
.AV "Are you Mr., Mrs., Miss or Ms. ? ",MS
.AV "Your address ? (three lines): Line 1:  ",ADD1
.AV "                              Line 2:  ",ADD2
.AV "                              Line 3:  ",ADD3
.AV "What is the date? ",DATE
.AV "You want to make a complaint about something you have
     bought  which doesn't work properly.
     Complete the sentence: `I bought a ....",PRODUCT
.AV "When did you buy it? ",DATE2
.AV "Where did you buy it? (name of company) ",CONAME
.AV "Type the company's address (three lines):  Line 1: ",ADD4
.AV "                                           Line 2: ",ADD5
.AV "                                           Line 3: ",ADD6
.AV "Write a word to complete this sentence:
   `Although it was all right at first,
    it now .... very badly.':",WRONG1
.AV "Write a word to complete this sentence:
    `It doesn't .... properly.'  ",WRONG2
.AV "A refund or a replacement?  Complete the following
     sentence:  `I want a ....",REP
.FI B:COMPLAIN.LET
```

All the commands preceded by full stops (eg .AV and .FI) are instructions to the program (hidden from the user), and will vary from one word-processor to another (this example was made using the WORDSTAR MAILMERGE program). The program takes care of formatting the lines to suit the varying lengths of the information supplied. After the student has answered all the questions, the final product is shown opposite.

This "poor man's Anthony Crispin" has none of the sophistication of the original. Nevertheless, it was prepared in about fifteen minutes, and can be used by any number of students to produce countless individualised letters of complaint. It could be extended to include size, quality, purchase price and so on; and a little extra effort in typing in more frameworks (perhaps FURIOUS.LET and DESPAIR.LET, in addition to COMPLAIN.LET), which could be "chained" to follow one after the other using the same data, would mean that each user could be provided with a range of letters, illustrating the language of polite complaint, the language of unrestrained anger, the language of bitter despair, and so on.



25, Trafalgar Street,
Hackney,
London, N.16.

29th September, 1985

Jones & Co.
12, Disraeli Place,
London, W.3.

Dear Sirs,

I am writing to complain about the wristwatch I bought from Jones & Co. on 10th September, 1985.

Although it was all right at first, it now loses very badly, and it doesn't wind properly.

As the wristwatch is still under guarantee, I am writing to ask for an immediate replacement and I look forward to receiving this as soon as possible. I am returning the faulty wristwatch under separate cover.

Yours faithfully,

Isabella Santos.
(Ms.)

Using the "search" facility

Another feature of many word-processors is the "search" facility, which enables the user to specify a word or phrase and move through a text viewing successive examples of the chosen item. Higgins and Johns, in *Computers in Language Learning* (Higgins & Johns, 1984), suggest the following exercise, using no program other than the basic word-processor and a selection of texts which have already been prepared on it for other purposes (perhaps reading passages or class handouts):

> A student, rather than be given constructed examples of, say, *if*-clauses, can be asked to use the FIND command in the word-processor in order to locate all the occurrences of the word *if* in a piece of test, and to try to state a rule which will account for the way the word is used.

Just as the Crispin program is superior to the do-it-yourself mailmerge idea, a purpose-made program could enhance the ordinary "search" function on the word-processor. The problem with "search" is that it will only search for the precise item you ask for. If you ask for "if", you may on some word-processors persuade it to look for "IF" at the same time, but it will not pay attention to "Had I done that, I would have . . ." although this is an equally valid way of expressing a condition.

One could envisage a program which, on being given the command CONDITION, would know enough about language to sort out

> If he had only done that . . .
>
> Had I only done that . . .
>
> Should she have done that . . .
>
> Were they to do that . . .

as representative of CONDITIONS (although it might be fooled by, for example, "Should she have done that?").

The same program might be taught to pick out examples of some verb tenses, passive voice, adverbs and the like, offering a menu of the available options to the user. The fact that it would undoubtedly make mistakes (would it recognise "fast" when used as an adverb? Might it identify "wicked" as a past simple verb?) would only serve to keep students on their toes, and prompt them to ask *why* certain cases presented problems while others did not. They could also sharpen their appreciation of the language by reading ahead and trying to predict where the next "search" command would land.

Using the "replace" facility

Word-processors which contain a "search" command usually also have its counterpart, "search and replace", which can be used in interesting ways. Take this sample text, quoted from an American book, *Computers and Reading Instruction* (Geoffrion, 1983):

> After the game, Billy runs home. He tells his mother about the home run he hit for his team. The team now has a run of ten games without a single loss. If their run of good luck continues, they will be the best team in town.
>
> Billy's mother says she is proud of him. At dinnertime she tells him to run into the bathroom to wash. She tells him to be sure to run the water over his hands to get all the dirt off. She also reminds him that tonight it is his turn to clear the table and run the dishwasher.

Although less than satisfactory as an example to use with adult learners in England, this passage will serve to illustrate a useful technique. The passage contains seven examples of the word "run", used with a variety of different meanings. The users' task is to find a suitable synonym for the first use of the word, in the phrase "Billy runs home". In the scenario presented in the book, a child suggests "race", and the teacher uses the "search and replace" facility to find *every example* of "run" and replace it with "race":

> After the game, Billy races home. He tells his mother about the home race he hit for his team. The team now has a race of ten games without a single loss. If their race of good luck continues, they will be the best team in town.
>
> Billy's mother says she is proud of him. At dinnertime she tells him to race into the bathroom to wash. She tells him to be sure to race the water over his hands well to get all the dirt off. She also reminds him that tonight it is his turn to clear the table and race the dishwasher.

The group then examines each occurrence of "race" in the text, and decides whether it is a valid synonym for "run" in the context of the story. The acceptable examples stay put, the teacher converts all the unacceptable examples back to the original "run", and the search is repeated to find other synonyms for "run" in its various contexts.

This exercise could be used with specially-prepared texts on any word-processor with a "search and replace" option. Once taught how to use the option, students could operate the program without assistance, and save and print a final version of the text, which could be used as a basis for class discussion, or simply commented on by the teacher.

The main differences between purpose-written programs and the use of a standard word-processor to simulate the same kind of effects is the convenience and ease of use which the dedicated program can provide, and its potentially greater degree of sophistication. The do-it-yourself word-processing versions are nevertheless adequate for adults to use, especially if the word-processor is simple to operate.

Audio linking

Computers are not limited to handling the written word, but with careful use can be persuaded to cope with spoken language, too. That is not to say that the standards of speech-recognition and speech-synthesis (the computer's ability to comprehend the spoken voice and make a vocal response) are sufficiently well-developed at the present time to be of general use in language-learning activities; but judicious use of an ordinary cassette-player in parallel with a computer can nevertheless help to bridge the gap between the written and the spoken word.

The cassette-player may be completely independent of the micro, operated by the student under a combination of instructions from the computer (on-screen messages like: "Now press the PLAY button on the cassette-recorder and listen to the text") and instructions recorded on the tape ("Now press the STOP button, and answer the questions on the screen"). Or, with very little trouble, the cassette may be linked to the computer so that it may be switched on and off under instructions from a computer program.

One way of using a computer and cassette-recorder in combination is the recording of a dictation on the audio-cassette, to run alongside a computer program which consists of a "correct" written version of the dictation of the plus a routine which will allow the user to type the text at the keyboard, and a further routine to compare the user's version with the "correct" version.

Even with this apparently simple set-up, there are complications. The user's input-routine should incorporate a text-formatter, so that words will not split across two lines of text. The comparison-routine must allow for the user's possibly idiosyncratic use of spaces, and not penalise it. These two factors are not difficult to manage, but it is surprising how many programs which require an extended input from the user fail to take them into account.

Most problematic of all, as with a student-oriented cloze program, there will be only one "correct" version of the text. It would require quite a sophisticated program to allow for variations which might legitimately occur — for example, dashes instead of brackets, or a semi-colon followed by lower-case as an alternative to a full-stop followed by upper-case. Even if the dictation were of the formal variety which included instructions about punctuation and capital letters, would the program allow for the use of square brackets instead of round ones? The user would either have to accept without question that the computer was infallible (always a dangerous practice!) or would be encouraged by the program to seek arbitration on points of divergence between the user's version and the accepted one. If the latter course of action were adopted, it is difficult to see how the computer scores significantly over the more traditional methods of giving dictations. Always supposing that you want to give dictations in the first place.

Note-taking skills

John Higgins and Tim Johns (Higgins & Johns, 1984) discuss some ways in which the computer may be used in the role of "demonstrator", where the user's role becomes the more passive one of reading, learning and inwardly digesting. One of the ways they suggest uses the audio-computer link-up to demonstrate note-taking skills: a passage is read on tape, and simultaneously a summarised version builds up on the screen, as if the computer were "taking notes" of the audio passage. Quite apart from its overt application of improving note-taking skills, such a program would sharpen the user's ability to pick out what is essential to communication, and what is peripheral to it. It would be useful to incorporate a facility for printing out the finished product at the end of the session, as an object-lesson in the construction and formatting of notes.

The next stage is to give the student an active role to play, and Chris Jones, a prolific CALL author, has produced such a program, although it is not a completely free note-taking exercise since "headings" are provided to steer the user in the direction of the required information. The program uses a technique often seen in textbooks, where the student is required to fill in a table or grid of information from details supplied in the listening passage. The advantage over the pencil-and-paper method is that the student can use the screen as a "scratchpad", adding or deleting information at will, until satisfied with the finished product. A "clean" copy could then be printed for the student to take away.

A further development of such a program might present a "blank sheet", allowing each user to type in a personal selection of notes in response to a listening passage. This could be a "real-time" exercise for a competent keyboard-user, or it may allow the student time for typing in between sections of listening, with the cassette under the control of the computer, so that, although the exercise would not be absolutely authentic, the audio passage would be heard once only, at a pace controlled by the computer (as it would normally be under the control of the speaker). The computer could then compare the student's version with a "model answer", looking for keywords, suggesting improvements where important ideas have been omitted, and even making an appropriate comment if the student included too much detail in the summary.

Reporting skills

The Kennings (Kenning, 1983) suggest an idea for a simulation which involves the user in passing on information to a third party:

> The learner can be asked to imagine that she is phoning her brother to pass on some instructions left for the two of them by someone else. For greater realism all sorts of details can be added and the task fitted into a story which gives rise to a series of activities.

Anticipating some possible objections to this idea, the authors continue:

> A purist might protest that a simulation on the lines suggested above lacks authenticity, that phoning implies oral communication, and that simulations ought to preserve the mode of communication of the situation being mimicked . . . There is no denying the desirability of making simulations as lifelike as possible . . . but taking liberties is something which is intrinsic to simulation . . . How many teachers would go to the trouble of always providing an actual telephone in a telephone simulation, or would avoid setting pieces of creative writing concerned with what, in real life, would be spoken messages.

There is always the possibility, in this instance, of linking a cassette-recorder to the micro, so that the telephone instructions are received aurally. Then instead of asking the user to pass on the messages to her brother, she might be asked to "jot them down" via the keyboard, as if leaving them on a notepad. Having once created a framework for this exercise, it would be a fairly simple matter to "slot in" a variety of different telephone-message situations, which might include, for example, an ambulanceman taking details of an accident from a distraught caller, or a householder making an appointment with the gasboard to have the water-heater repaired.

Special education — contributions to CALL?

It is worth investigating every possible source for programs which may be borrowed without alteration or for ideas which can usefully be adapted. One of the sources which may have much to offer in the future is the body of programs in the field of special education and education for the disabled, since many of these rely on simplified input routines, and input devices other than the keyboard — for example, concept keyboards, and joysticks and switches programmed to make simple two-, three- or four-way choices.

HONEYTALKER is one of the few speech-synthesisers which could be usable for language work. The main drawbacks of most speech-synthesisers are that they are difficult to understand, and speak American English. HONEYTALKER, intended for severely disabled users rather than for home-computer use, is of a much higher quality, and can be programmed to speak "real" English.

A suite of programs for older children with reading problems has already been mentioned, and some of the programs in this suite can be easily adapted to suit an adult audience.

Another source of ideas lies in programs for the deaf. TIMECODE is a system for subtitling video film, which, in addition to its original purpose of adding subtitles to standard film footage, might be used for captioning, say, video stills to make versatile flash-cards, which may be displayed with or without captions.

There may also be a use in ESL/EFL for a system used with deaf speakers which compares the "voiceprint" of the speaker with a "standard voiceprint" of chosen sounds or sound-groups, displaying the comparison visually, and allowing the user to repeat the performance to get a closer pronunciation-match.

These few suggestions will suffice to give some idea of the scope of work in progress for special education which may be of value in language applications. But the traffic of special-education programs need not be a one-way system. While language-learners may benefit from ideas from special education, it may well be that programs designed for adult literacy could also find a further application in the teaching of the handicapped.

Databases

The computer in the classroom is always a means to an end, and must not be made an end in itself. One kind of program which illustrates this perfectly, and which should be a part of any language software collection, is a database program, since this can not only be used to find out how databases work, but can generate a lot of language activity, both round the computer and away from it. Derrick Daines' *Databases in the Classroom* (Daines, 1984) provides a useful introduction to the subject.

Small groups of students might be encouraged to set up simple databases on any topic which can be suitably evaluated and on which they can build up relevant information. Information-gathering could involve the introduction of a variety of research methods, including a visit to the local library. Topics might be anything from details of the pop-music industry (which group recorded which song? when? how far did it get in the charts? what else was on the record? which records have been published on this label since 1982?) to schemes for identifying makes of car or issues of postage-stamps.

Database project-work

A comparatively simple database like MICROSCAN could be used for a large-scale project involving a whole class, which might include:

☐ discussing and deciding on a suitable topic for a survey which may be as simple as a traffic survey in some nearby streets, or more comprehensive, such as a survey of local employment prospects or housing conditions;

☐ deciding how big a sample should be taken, and how the sample should be obtained (within the class, within the college or institute, from members of the public and so on);

☐ formulating suitable questions for the survey, and making a "dry run" within the group to see if any questions need amendment;

☐ discussing how the results of the questionnaire will be analysed, what "fields" or headings should be created in the database, and what kinds of information the database will be required to provide;

☐ amending the questionnaire in the light of the previous two stages;

☐ using a word-processor to prepare a model questionnaire, and perhaps duplicating it, if facilities are available to the students (otherwise having it duplicated by whatever means are provided);

☐ carrying out the survey;

☐ creating the appropriate fields (that is, topics on which information has been collected) in the database and entering the results;

☐ questioning the database to obtain the desired information and print it out;

☐ writing up reports of the findings, either "straight", or as "newspaper articles", and perhaps incorporating results in the form of graphs and charts;

☐ if the subject is suitable, holding a debate on the findings, in which various individual interpretations may come to light, and suggestions be made as to further action to be taken as a result. The debate might even be an event to which an "expert" could be invited to be questioned by the group and faced with the results of the survey.

Only two of these activities actually involve the use of the database, and yet it is doubtful whether any serious project of this kind would be carried out effectively without it. The database certainly speeds up collation and analysis of results, and often reveals unexpected relationships between the classes of data.

The importance of databases in language-work is this: before information can be taken out of a database, it must first be put in (the exception being the public databases such as Prestel). While future generations of students may wish to question databases created by previous students, the real value of a home-produced database to language-students is in assembling and discussing the data before keying it into the machine, and this can be a rich area for project work, either for the whole class working on aspects of a single subject, or for groups handling individual topics.

Communicating with the outside world

Databases do not have to be made from scratch. A telephone-link to a public database, such as Micronet or Prestel, can give access to vast quantities of information which would reward exploration, plus the added bonus of a "local viewdata system", which allows users to create and edit their own database — an electronic "house-magazine". Prestel is the "national database"; Micronet is a "closed group" for computer users, which occupies part of the Prestel system. Micronet subscribers have automatic access to Prestel, but membership of Prestel does not automaticaly admit you to Micronet.

Not only does Prestel provide facilities for students of computing and electronics to examine a genuine working database, it also opens up a range of opportunities for teachers in other subject-areas — not least of which is language-work — to benefit from vast stores of information on almost every imaginable topic, and exploit them in whatever ways they choose.

The Prestel database contains a wealth of material under the direct heading of "education", including details of courses and learning opportunities, and sections on qualifications and careers advice. A "School Link" database within Prestel is to provide an extended range of educational services, including programs which can be loaded into the user's own computer and saved on disc or cassette for future use. (This kind of service is already widely available through Prestel and other databases, but School Link will cater specifically for educational users). There are also special educational services, like the Interactive Numeracy Project, which is developing CAL materials for basic numeracy.

Possibly of even greater value for language-based activities is the ability to explore what is available in the hundreds of subject-areas which are not directly related to education. News, weather, traffic information and timetables are among the more mundane subjects which can be accessed through a public database. But there are also enormous banks of data which can be found by "browsing" through the thousands of pages on offer. There is a quiz based on knowledge of the London Underground system, others on aspects of English literature, a section on the Language of Politics, sections describing the principal features of most countries in the world, geographically and politically — the possibilities are endless.

The Council for Educational Technology (CET), to encourage the exploitation of Prestel as an educational resource, has made a simplified classified index of useful subject areas "under 66 subject headings from archaeology to World Studies", and also a range of learning materials (for teachers and students) to support the use of Prestel across the curriculum.

Electronic mail

A telephone-link also provides opportunities for "electronic mail" — direct and fast links with other micro-users, which allow messages to be sent to, stored in and collected from "electronic postboxes". Although this feature is not much used in the educational world at present, the possibility exists of computer-links between educational establishments, which could lead to improved contacts, greater communication and mutual co-operation among staff, or even to "pen-friend networks" of students in different parts of the country. Already, any school or college can apply for a personal "frame" in Micronet which they can use to advertise their activities, separate Micronet areas being set aside for Primary, Secondary and FHE sectors.

As these connections are made via standard telephone-lines, there is a question of cost to be considered. Once the initial connection has been set up, there are additional charges — the cost of telephone calls, and a charge per page of information accessed. The good news is that for almost every user, the telephone charges are those of local calls, and there are many thousands of database pages which may be accessed free of charge. There is also software available (directly downloaded into the computer from Prestel) which will tell you when a call-unit is about to finish, so that you can if you wish leave Prestel without incurring extra call-charges. The user is also told of any page-cost in advance, and so can decide whether or not to go ahead and look at the page in question. An additional bonus is that, to avoid logging on to the system during the day, when call-charges are high, teachers can browse through the pages during off-peak periods, and save the pages they want on disc or cassette for future use, so that students can examine them at leisure without any further expense. Even with care, however, considerable personal discipline is called for in order to control the cost.

Which way for the future?

Our present view of microcomputers is narrow and limited. Narrow, because, when it comes to computers, we are still inclined to think in terms of drills and exercises, instead of creative discussion and creative writing.

Limited, because the microcomputers available to us today are tiny compared with those which will become the norm within the next few years.

The programs being used for language activities today will soon become museum-pieces. What will take their place?

The programs of the future *must:*

- ☐ integrate the computer with other classroom activities, making it a classroom tool in the same way that a cassette-recorder is;
- ☐ promote group activities, rather than solo efforts, and ensure that communication among the users is the principal concern, rather than communication between the students and the machine;
- ☐ show greater sensitivity to input by making intelligent guesses at meaning, rather than saying "I don't understand you";
- ☐ make allowances for the fact that users may sometimes be neither right nor wrong, but somewhere in between;
- ☐ allow a much greater degree of "natural language" exchange, both in what the user can put in, and what the computer gives back;
- ☐ encourage questioning rather than blind faith in the infallibility of the machine.

Chapter 3

Making CALL programs "user-friendly"

Elaine England, of the University of Wales Institute of Science and Technology, contributed a searching article to *Educational Computing* (England, 1983), in which she posed the question: "What does it mean to be computer literate?", and then went on to highlight essential differences between "print literacy" and "computer literacy". These differences can be used as springboards for a consideration of ways in which CALL should be introduced to students.

Successful book-reading strategies involve "implicit background knowledge" about the organisation of information: we expect to find books arranged in the usual, predictable way, and as we gain experience in reading, we develop methods for obtaining necessary information quickly and easily, based on our expectations of overall layout.

Drawing parallels between the use of books and the use of computer programs, Elaine England continues:

> Sometimes, when one starts a program, there is a sense of not knowing where one is in relation to the rest of the material: how long the program is, how long you need to pace yourself and your concentration, how to check back on previous information and, very often, how to stop the interaction. All these are achieved easily with printed material and are taken for granted . . .
>
> Screen layouts vary considerably from use of separate boxes located at different parts of the screen, part screen deleted or retained, animated graphics, highlighting, flashing, etc. New reading strategies are needed to make sense of these. Often a flashing cursor at the bottom of the screen draws attention there first, to the question rather than the information.
>
> Books allow forward and backward movement, checking and re-checking of information. Many programs are linear forward moving only. The student is allowed only one chance to absorb the information...
>
> There are psychological barriers which arise from the change of medium. Apprehension occurs because of lack of knowledge, fear of going wrong, breaking the machine, what to do next, and this affects the interaction.

Although some of these arguments are more applicable to "programmed learning" exercises than to the type of material now being developed for CALL, the concepts behind them are still of major importance. The Further Education Unit has declared that *all* students have a right to training in computer skills, and for many students this will mean familiarising themselves with the operation of a computer via programs of direct relevance to their course-material. This places a responsibility on programmers to provide software which will attract and hold the user's attention, minimise the "culture shock" of moving from the print medium to the micro medium, and allay any fear or suspicion which may be felt about learning to learn through the new medium.

To summarise the points made by Elaine England:

☐ computers require different learning strategies from those required by books;

☐ one of the stumbling-blocks is the absence of the familiar points of reference found in books;

☐ fear of the machine may lie behind some antipathy to CALL.

Some cold comfort may be found in the indisputable fact that many students have never acquired adequate learning-strategies in *any* medium — perhaps for the reason given in the article, that "training in accessing information from books is usually haphazard and assumed through exposure" — and for those students, computer skills may be planted in virgin soil. The lesson from this is that, unless a conscious effort is made with computer literacy, many people will fail to acquire these skills, too, and that will be an even worse handicap in an increasingly technologically-minded society.

Careful presentation of CALL programs can go some way towards providing a frustration-free environment, in which students can acquire computer skills alongside their language skills, without one dominating or impeding the other.

There is no reason why computer programs should bear any resemblance to books. Nor is there even a remote possibility that all CALL programmers will pursue a common design policy, so that computer programs could eventually develop the kind of superficial resemblances to each other which one book bears to another. So how can students be helped to acquire a new learning strategy from a mass of diverse, disconnected software? And how can language teachers learn to assess the quality of a piece of software, not only from the standpoint of its language-learning potential, but also from the point of view of the quality of programming techniques?

Program assessment

"Garbage In, Garbage Out" is a well-known saying in the computer world — not as elegant as Coleridge might have put it — "we receive but what we give" — but to the same point. English teachers may not appreciate the fact that a program is only a reflection of the programmer, but some of their caution about bringing computers into the classroom is based on their perception that most of what's out there is inadequate. They're right. Here's an industry leader, quoted recently in *Money Magazine*: "There are tons of software out there. Much of it is pathetic." Programs don't work properly, don't do the stated job, and can't be understood from the accompanying manuals.

The above opening to an article by Dr Joan Baum in an American magazine, *Clarion* (Baum, 1983), may be applied equally well on either side of the Atlantic. Dr. Baum goes on to make a valuable point:

> If informed teachers of English become reviewers of software, they may induce modesty in would-be programmers and prevent further glutting of the market.

Program-assessment does not call for much knowledge of programming. Assessment of the pedagogical value of a program needs no programming knowledge at all; assessment of computer-worthiness demands a little knowledge, but it is knowledge worth acquiring, in order to avoid what Papert calls "the force-feeding of indigestible material" to students, and to prevent expensive mistakes.

The "usability" of programs varies so widely, from abysmal to impressive, that it is important for prospective buyers to have a clear idea of how much computer software can be expected to do. The object of this chapter is to outline some of the desirable principles of programming which teachers should look out for in the programs they buy for CALL, and to offer warnings about some of the poor programming techniques which may still be found in abundance. The chapter should provide enough information to program-buyers to enable them to distinguish between the sheep and the goats of CALL software; it should therefore also help them to filter out some of the least suitable materials, so that students' "new learning strategies" are built upon the best examples of their kind; and if it incidentally gives some program-*writers* a fresh idea or two, so much the better.

Documentation and program instruction

The first contact a user (whether it be a student or teacher) has with a program is usually through its documentation. Most computer magazines contain reviews of products, but very few of these relate to educational programs, and few of the educational programs relate to language-work. *Educational Computing* is the honourable exception, since it contains regular reviews of educational software of all kinds. If a review is unobtainable, then — unless the supplier is willing to allow experimentation with software — the paperwork may be the only chance of assessing a product before purchase.

There are two kinds of instructions associated with computer software. One is the printed documentation, the other is the instructions on screen before or during the running of the program.

Printed documentation

The value of printed documentation is inestimable. It can take various forms, ranging from a mere list of basic operating instructions, via hints on operating strategies and techniques, to wide-ranging suggestions for exploiting the software by integrating it with other classroom activities.

Be cautious if the documentation is sealed up with the software. Such is the fear of "software piracy" that computer programs are rapidly becoming the only product on the market which the purchaser is expected to buy "blind", on the basis of the title alone. You may not be able to try out the program, but the least you can expect is a sight of the documentation.

Documentation in printed form may include some or all of the following:

☐ a name and address or telephone number to contact in the event of trouble with the program — or simply because you have a suggestion for an improvement;

☐ necessary information about what configuration of hardware is required to run the program;

☐ information about the program-files to be found on the disc or cassette (so that the purchaser can check that all the necessary bits and pieces are in fact present);

☐ details of any installation procedures required prior to running the program, including how to configure the program for a particular printer, if necessary;

☐ information on how to make a back-up copy of the program for safety purposes (although this is not always possible);

☐ details of the object of the program, with a summary of its main features;

☐ samples of typical screen layouts and/or printouts, and samples of typical user input;

☐ details of the input required from the user, including important factors such as whether or not the RETURN or ENTER key needs to be pressed following an input, or if any key is reserved as an "emergency exit" from the program (often the BREAK or ESCape key is used for this purpose).

If the paperwork contains all or most of these features, the program itself will *probably* be well-devised and unlikely to cause problems. On the other hand, glossy handbooks do not necessarily mean glossy programs, and documentation that looks expensive is no guarantee of a satisfactory working product.

On-screen instructions

If there is no printed documentation, the program is not necessarily a bad one — it may be that its use is so transparent that paperwork is unnecessary, or that instructions for use are incorporated within the program. In either case, make sure that the answers to the first five points listed above are printed on the package.

The incorporation of instructions in the program can be a blessing or a curse, depending on the circumstances. The blessing is that on-screen instructions can not be lost. But the curses can be many and various. You will need to know

☐ if the instructions can be by-passed by users who already know the program and have no further need of them. It can be very irritating to be obliged to page through masses of instructions every time the program is loaded (though a single screenful is acceptable);

☐ whether there are several "pages" of on-screen instruction, and if so, whether the user can *page backwards* to recap forgotten or misunderstood details before starting the program, or alternatively whether the whole instruction-phase can be repeated as often as required;

☐ whether instruction-screens can be recalled if necessary while the program is in use, or alternatively whether a summary of the program commands is available, either on demand or as a permanent feature on the screen. It can be very frustrating to users if they have to re-start the program from scratch because they cannot remember a program command;

☐ whether, if HELP is available during the course of the program, the HELP command is displayed on screen at all times.

Menu-driven discs and programs

The next stage after digesting the printed documentation is to load the program (or programs, if you are using a disc which contains several different activities).

Programs are much easier to understand and use if they are "menu-driven" — that is, the programmer has provided users with a simple way of progressing through the programs on a disc, by choosing the next course of action from "menus" of the options available. The following list covers some of the features to be looked for in menu-driven programs, and some of the pitfalls:

☐ if a disc contains more than one program, there should be a menu of options, consisting of each program-name plus a short description, allowing selection by typing the number or letter which appears alongside the user's choice. This avoids the need for users to remember a program-name (which may be something like "FCE2.COM" or "PHRAS.BAS") and remember the command sequence necessary to load it correctly;

☐ if the menu offers a choice of programs, or a selection of texts or subjects which are held as data-files on a disc, check that the menu consists of "real" names, not just computer file-names called up from the disc: it is far easier to imagine the contents of "A Day in London" than "DAYLOND.DAT";

☐ when a program has been selected and loaded from the main menu, further menus may be used to set up options within the program; these may be as long or as short as necessary, perhaps offering a choice of twenty different texts to work on, or possibly just a choice between a fast game or a slow one. There may be several menus to work through before the program proper can begin, but they should be simple enough to get through quickly, and may involve some language-work in themselves, especially if it is a group-project;

☐ if the program involves the user in choosing a text or file to work on, check that there is in fact a menu of texts to choose from; it is very frustrating to load a program, and find it asking “Which text do you want to use?” when the only way to find out what the options are is to jettison the program and examine the disc directory. If there is no menu of texts, be sure to provide a printed menu which can be kept with the disc. The printed menu should consist of the *exact* file-name of each text, along with a brief description of its content;

☐ at the end of the activity, another menu should give the option of repeating the activity, perhaps repeating the activity with different parameters or different data, or exiting the program;

☐ when the user chooses to finish one of the activities on a multi-program disc, the program should “chain” back to the original menu, allowing the choice of another program on the same disc if desired;

☐ one of the options on the main menu should be a “FINISH WORK?” option, to allow the user to exit the disc. Otherwise the only way of escaping is to “pull the plug” on the machine.

Once a menu-driven disc is loaded, users should be in a closed loop, the purpose of which is to lead them through a series of choices in a simple and unambiguous manner, so that chosen programs can be run without difficulty, before exiting the loop gracefully via the “FINISH WORK?” option of the main menu.

Program security

Once loaded, programs should be, but are not always, reliable to the extent that:

☐ it should be possible to press any key on the keyboard at any stage without “crashing” the program. This includes

> pressing alphabetic keys when numeric input is expected (e.g. typing “four” when “4’ was required);
>
> pressing control-key combinations during an input;
>
> using upper-case for lower-case and vice-versa;
>
> inadvertently pressing escape-route keys (the program should check whether you really want to finish before irrevocably coming to an end);

The worse that should happen in any of these cases is that you get a polite message reminding you of what is required, or alternatively, that the program simply does not respond to an “illegal” input;

☐ programs should not produce incomprehensible messages in *any* circumstances, and should not dump the user unceremoniously out of the program without warning or explanation if a program-failure occurs;

☐ if any part of the program is missing from the disc, or if the program has been tampered with in any way, the user should be informed that the disc is incomplete or corrupted, and then the program should continue, if this is possible, or make a graceful exit, after a suitable explanation has been given in plain English.

Checking input

One of the easiest ways to check how secure a program is in use, is to feed it with unexpected responses: if you want to choose Option 4 from an menu, and the program expects the figure "4", try typing "four" instead, or if you want to select Option B, type "b". This may seem unnecessarily malicious, but remember that this is the kind of error which first-time users will make entirely without malice, and in order to protect them, you need to know that programs will stand up to reasonable misuse.

The same rigorous standards of security should apply to everything which is typed at the keyboard in the course of the program.

Tolerance of input

Of course, the fact that a program is technically perfect, and does not come to a halt when it meets an unexpected response, does not necessarily mean that it works properly. It may do everything the programmer meant it to do, and yet still be unsatisfactory to the user.

A fault which occurs even in quite sophisticated programs is the "untrapped case", in which the program expects either upper or lower case inputs from the user, and refuses to accept as correct an input which is in the "wrong" case. The program does not crash — it merely tells the user that the answer is incorrect. This is an unpardonable fault in programs for those whose confidence in their own use of the English language is easily shaken, and who are far more likely to believe themselves in the wrong than a computer. It is even more unacceptable, in that this is a very simple check to make, and often involves no more than an extra line or two in the program.

On the other hand, programmers sometimes go to the other extreme, and check only the first word, or even the first *letter* of an input. Provided the input begins in the right way, any old rubbish may follow, and the program congratulates the user on being correct. This is done on the basis that, if a certain proportion of the input is correct, the likelihood is that the rest will be correct, too.

Robert L. Burke (Burke, 1982) writes about "tolerance" and "flexibility" in educational programs:

> . . . knowing that the correct first and last letters of the answer have been given is often enough evidence to conclude with a high degree of certainty that the answer is correct.

Whether or not this may satisfy the teachers and the taught in other disciplines, it is not an adequate approach in programs for language-learners, though there is a case in some applications (those in which correctness is less important than the effectiveness of communication — like, for example, simulated dialogues) for basing input-checks on some method of keyword-matching, in which the computer is satisfied as long as certain keywords appear in the course of the input.

Bearing in mind the modern primary-school technique of correcting only the most blatant of spelling errors, there may be an argument for allowing certain programs to overlook the spelling, so long as it is recognisably similar to the accepted form. It is quite possible, however, to construct a program which will make "silent" corrections of spelling on screen.

Input-checking for CALL

Checks on input in CALL programs should be of the highest standard. They must be systematic and thorough, and should take account of the following, where appropriate:

Programs SHOULD

☐ give the user clear indications of what kind of input is required, and how it should be terminated (for example, "Please type your name, then press the RETURN key:");

☐ ensure that the user's input is converted to the same case as the "correct answer" data in the program;

☐ in an extended input, either explain the error or mark on the screen the point where the first error occurs; this is a minimum requirement — it may sometimes be advisable to point out spelling mistakes, if spelling is the only error in the input;

☐ at all times use forms of English which are acceptable to native English speakers, and demand the same of the student.

Programs SHOULD NOT

☐ mark a case-error as "wrong", unless upper/lower case is of importance — say, for days of the week, or at the start of a sentence, and even then, the program might simply point out the error rather than imposing a penalty on it;

☐ penalise over-use of spaces between words, or penalise missing spaces, except where this affects communication;

☐ remove the faulty input from the screen until the user has had a chance to check up on *how* it was wrong — a program which removes the input and then says "Try again!" runs the risk of trapping the user in an endless loop, from which the only exit is to "pull the plug" on the machine.

When the user's input is substantial — a word or more in length, rather than a simple multiple-choice answer — it is not usually enough for the computer simply to say "this is wrong", because there are so many ways of being wrong. It is unacceptable to treat a missing punctuation mark or capital letter in the same way as a completely erroneous answer, and it is quite unforgivable to mark an input as "wrong" because the student typed two spaces between the words instead of one. As recently as, say, 1982, memory constraints on microcomputers might have excused poor input-checking, but this is no longer the case. It should be a matter of course to eliminate excess spaces, and check for missing punctuation or the wrong case, and it does not take a very sophisticated program to check if the principal fault in an input is one of spelling, and either take appropriate steps to bring this to the user's attention, or simply ignore it.

An unnecessary distraction?

An argument used in defence of poor input-checking is that the pointing out of minor errors such as missing capital letters "distracts attention from the object of the exercise".

The other side of the coin is exemplified by a (commercially marketed) program for EFL learners in which the stock response to an input error was "I don't understand" — regardless of whether the error was the result of ignorance of the machine, ignorance of the English language, poor spelling, putting two spaces between words instead of one, or simply not knowing what kind of input was required. Since the student's input was wiped out before the appearance of the error message, this led to frustration on the part of the user, who had no idea what the error was.

On the other hand, the same program tolerated various types of linguistically poor input, such as not requiring question-marks at the end of questions, and not demanding capital letters for names, days, beginnings of sentences and so on. This led to the curious situation in which "john arrived on tuesday" would be a "correct" response, but "John arrived on Tuesday." would be "wrong" because of the extra space between "arrived" and "on".

The pointing out of minor errors need not be made into such a big issue that it "distracts attention from the object of the exercise". Case-errors may simply be corrected by superimposing the correct form of the letter in reverse-video (say, blue on white, instead of white on blue) over the user's own input, and missing punctuation might be inserted in the same way; metathesis could be corrected by a simple animation routine to switch the letters into the correct order, and missing or extra letters inserted or deleted as appropriate. All these points may be handled "silently" — without any comment — unless the object of the program is to improve accuracy, in which case they may be made a feature.

The view has been expressed that teachers "must allow students the freedom to *choose* to be wrong", and that pointing out their errors may be a serious infringement of their liberty. The fallacy of this attitude is that *unless the student knows what is right,* there is no real "choice" to be made. Students, especially in the early stages of learning a language, tend to lean on printed materials far more than they should, and often regard the printed word as infallible. This tendency may even be exaggerated when the medium is a microcomputer. There is a real danger in allowing seriously non-standard forms of English to go unnoticed, in that some students may believe that, because the computer accepted their input, it must be "good English".

John Higgins, speaking at an English Speaking Union seminar (November 1982), referred to programs which allow the user to "turn the language upside down" in order to examine the rules under which it operates. I am sure he would be the first to admit, however, that you cannot consciously turn something upside down until you know which way is "up", and input-checks can act as signposts to students who are not yet sure of the "structural orientation" of the language.

Responding to incorrect inputs

Hints and clues

Unlike a book, in which the answer to a question is either given or not given, the computer can drop hints of various kinds before the final revelation of the answer, and this may be a great help to students who need coaxing. Also unlike a book, the computer can give assistance exactly suited to the student's needs.

Suppose the exercise to be one in which the student is required to find a word to complete a sentence (that is, gap-filling!). The program might start by offering a gap of indeterminate length, and if this proved insufficient, could then go on to indicate the precise number of spaces in the gap, then offer the "shape" of the word as a further clue (tall letters, squat letters, letters with descenders), and finally give the initial letter as a clue, before revealing the whole word. A textbook, if it offered format-prompts at all, would select one of these options, which may provide the student with too little information, or more than is necessary. The computer can offer any or all of them, or none, depending entirely on how that particular student copes with that particular example.

Appropriate responses

Once errors in input have been detected and identified, the program must decide on a suitable response. Context is an important factor in the handling of user-errors: a method of error-handling which may be appropriate in one situation may be wholly inappropriate in another. For example, the use of capital letters may be merely incidental to a vocabulary program, but of the essence in a program on the use of punctuation.

A good measure of appropriacy is the teacher's own classroom technique. Students should not get away with errors on the computer that the teacher would not let them get away with in class. On the other hand, the program should accord no more importance to a particular error in the context of a program than the teacher would normally give it in the classroom.

Another criterion of suitability is the environment in which the program will be used. The standards required of a program which is to be used without the supervision of a teacher will necessarily be much higher than those applied to programs for whole class plus teacher use. Any program which requires an extended input and does not check its structure or spelling, and respond appropriately, will probably require supervision, unless there is a printed output which can be checked later if necessary.

The way user-input is acknowledged is a thorny problem. Most teachers of adults, and many teachers of children, reject the smiling/frowning face approach to indicating rightness or wrongness. Most adult *students,* as well as their teachers, find such "rewards", including music and/or audible "raspberries", patronising and unacceptable. A program which plays "Rule Britannia" in response to a correct answer and the "Dead March" when the answer is wrong (and such a program does exist) will not be taken seriously and in all probability will be rejected outright by students in search of educational material. Furthermore, as has already been indicated, the user's response may be neither right nor wrong, but something between the two, in which case a simple explanation or perhaps an arrow pointing to an error would be more appropriate.

The general opinion amongst teachers and users alike seems to be that responses are functional and should be treated as such. "Fun" responses are only fun at first, and after that the only requirement is to know which answers are right and which are wrong. In some cases, "fun" responses may actually be damaging, in that they consist of increasingly sarcastic gibes at the user's lack of success, which is often a feature of commercial computer-games and, unfortunately, of some "educational" software.

Possibly the best approach is to treat correct inputs with respect, without "going over the top" in uncalled-for congratulation, and to treat incorrect inputs with even more respect:

☐ in certain circumstances, giving the user the option of a hint or clue *before* a response is attempted, as well as after;

☐ letting the student know if there is an error, without treating errors unduly harshly;

☐ offering a hint or clue, if appropriate, to help with a further attempt, and making this hint different from any pre-attempt hint;

☐ making "silent" corrections of small errors which do not affect the overall acceptability of the response;

☐ offering an "escape route" for students who simply do not know what response is expected;

☐ *never* putting students into an endless loop, on the principle that "they'll get the right answer if it kills them".

Scoring

The use of scoring as a mechanism for indicating degrees of rightness depends on the context and application of the program. In a competitive game program for EFL or ESL, some kind of scoring will be expected by the user. In a literacy program, however, the urgency of pressing on with a program in order to improve the score could be wholly inappropriate, and the sense of failure which might result from a poor score, especially in comparison with a "highest score today" feature, would defeat the object of the program.

One element of scoring which is often overlooked is the "final account" given when the user exits the program. In any other branch of education, "You answered 1 questions and had 1 questions wrong" may be regarded as an amusing peccadillo on the part of the programmer; in language-work aimed at EFL/ESL students, it is an unpardonable sin.

Colour, graphics and sound

Three features of programming which may sometimes be passed over as only subsidiary or minor elements in language-work can in fact be put to good use, and these are colour, graphics and sound.

Colour

Although a program may do a good job in monochrome, its appeal will be enhanced if a colour version is used. Colours need to be bright without hurting the eyes, and there should be contrast without disharmony. Some programmers seem to place emphasis on an attractive appearance at the expense of readability (for example, mid-blue writing superimposed on a dark blue background), but for the satisfaction of a greater number of users it is wiser to ensure that there is enough contrast between "paper" and "ink" to make reading easy. White writing on a strongly-coloured background is a combination which works well.

An important aspect of colour which is often overlooked by programmers is that two colours which look startingly different can have the same "grey tone", so a colour program run on monochrome equipment may not only lose the effectiveness of the colour display, but can sometimes be virtually unusable because the contrast between "paper" and "ink" has disappeared.

There are two sides to this story. First, let the buyer beware! Most programs nowadays are written for colour display, so if they are to be used with mono equipment, ask for a money-back guarantee from the programmer/supplier that they will be equally effective in mono. Secondly, it behoves programmers to see that their programs run effectively in both forms before they offer them to the public.

An interesting possibility is featured in a typing tutor program, ALL FINGERS GO, and also in computerised SCRABBLE, where the colours can be set by the user to suit personal tastes, which means that a viewer with a mono display can set grey shades which will look good in mono even if they clash horribly when viewed on a colour screen. This kind of facility may also be useful to a colour-blind user.

Graphics

The ability to display pictures is not the most important aspect of a computer's graphics capabilities. One important element of graphic design which is often overlooked is the appearance of the ordinary letters of the alphabet on the screen. It is still quite possible to find micros which have only upper-case characters, and there are also machines on which the lower-case characters have no "descenders" (that is, the g, j, p, q and y characters "sit on the line" instead of extending below it), which severely hampers legibility. Many micros have very "chunky" characters, so that reading the screen is like reading a page of teletext, which not only limits the amount of text on the screen, but can also be rather unpleasant for extended use. It is therefore well worth checking on the style and readability of the screen display before buying a computer.

What most people regard as "graphics" however, are the micro's picture-drawing capabilities. Drawing pictures on a micro — even with the aid of a special graphics program — can be a time-consuming business, though often fun for the programmer and sometimes spectacular for the user. The quantity of picture-drawing used in educational programs must therefore represent a trade-off between the programmer's time and the value to the end-user.

However, the drawing of pictures is not the only way in which graphic design can be used in computer programs, and some of its other uses are potentially more valuable in adult education. The three principle uses are:

☐ to catch the attention;

☐ to illustrate features which may be expressed more readily in pictures than in words;

☐ for decoration and rewards.

Attention-catching

The first of these uses is essential, and includes the use of colour, shape, space, contrast and size, as well as conventional picture-drawing.

The reader of a book is aware of the changes brought about as the pages are turned, because the page-turning process is entirely within the reader's control. The micro is like a book with only one page, and that page is not wholly under the control of the "reader".

In one sense, the micro is more flexible than the book. A book is either turned to page X or page Y; a micro can show the whole of page X, and then gradually superimpose page Y, altering first one part of the screen, then another, until the whole display has changed. The snag with this is that if the "reader" should glance briefly away from the screen or blink at an inopportune moment, an alteration in the contents of the "page" may occur so quickly that it goes unnoticed.

The programmer can actually *help* the user by exploiting the ways in which computers are unlike books, using colour and graphics to complement each other as methods of attracting attention to certain portions of the screen, so that changes in the display do not go unnoticed.

Attention-catchers are considerably more versatile on the screen than in a book. In addition to underlining and "boxing", the computer can call attention to points on the screen by reverse video, by changing the colour of the appropriate area completely, flashing the information (or the colour) on and off, or making one portion of the screen brighter than the rest.

Another advantage of the micro is the possibility of presenting material in manageable chunks, rather than in the form of whole "pages" at a time. A screen display can be built up in stages, thus drawing attention to its elements as they appear.

Because computer programs do not rely on expensive paper to transmit their messages, they can also afford to be more expansive in their arrangement of the "page". If it makes sense to display a small amount of information at any time, even a single word, a computer program can do it, whereas a publisher might jib at the waste of space. Screen displays can therefore be simple and uncluttered, which allows the attention to be drawn to essential elements more easily.

Many computers now provide a choice of screen displays or "modes", the common ones being 40 or 80 columns across the screen. The choice of display will depend on the application, but generally speaking, since there is no economic reason for squashing information into as small a space as possible, more displays containing larger characters are more acceptable than fewer displays packed with small characters. Even when texts are involved which may occupy more than one screen, it is usually more satisfactory to display them in large characters, and allow the user to "scroll" up and down through the text, selecting the portion to be displayed on the screen, as though the screen were a "window".

Concept illustration

The second use of graphics, for the purpose of illustration, is equally valid. This may take a very simple form, as in the TWO STICKS program, where the graphic representation consists of two block-lines of differing length, used to illustrate comparison of adjectives. Or it may be more complex, as in the PHOTOFIT program which uses face-drawing to assist with the use of comparatives. Illustrating concepts may not involve "graphics" in the accepted sense at all, but might employ animation techniques to move text around the screen, for example in a program concerning word-order or jumbling. In each of these cases, the graphics serve a purpose other than simple picture-drawing. The drawing of complex pictures which are merely prompts for vocabulary work can sometimes be sheer self-indulgence on the part of the programmer.

Decorative graphics

The third way of using graphics — for decorative purposes — is less easy to justify in a cold economic climate. On the other hand, it is sometimes difficult to draw a strict line between graphics which assist the user and those which are mere decoration, and the extent to which good graphics are an incentive to adults in their use of a program is largely unknown.

The pro-graphics camp would argue that a program with sophisticated graphics looks more interesting and therefore holds the user's attention longer. The anti-graphics camp take the view that adult language students either *choose* to study or *need* to acquire language-skills, and this is sufficient motivation to use a program.

There is also a case to be made that, in a world of scarce resources, the programmer's time would be better employed in writing another program than in providing superfluous graphics.

Sound

The sounds made by computers themselves (other than when they are linked to audio equipment) are generally unsophisticated, and largely unnecessary in terms of educational programs for adults. Computer sound includes noises which draw attention to an activity, or delimit activities, or perhaps indicate approval or disapproval, as well as musical introductions and rewards. One factor which is of paramount importance in all of these applications is environment.

In a classroom, sound can do damage in two ways. In the first place, it may disturb the work of students who are engaged in other activities. Secondly, a program which produces an audible "raspberry" when an error is made draws attention to the error — and consequently to the student who made it; few students persist in using a program which proclaims their faults loudly to their peers, with the result that the program may be used only by students who are already competent in that activity, which rather defeats its object.

It is a simple matter for the programmer to provide a facility within a program to turn the sound on or off, and this is a feature which should be incorporated as a matter of course. CALL programs would hardly suffer in content if sound were banished completely. As with the expectation of colour, however, modern television and home-computer games teach us to expect a sound accompaniment, and to think that we are being shortchanged if we do not get it — even if the only thing we do with it in the classroom is to switch if off.

Printers

In some computer activities, the task is essentially ephemeral, and any odd words of interest can be jotted down by the user in the course of the program. Other activities involve text or examples which may profitably be taken away for later study, and sometimes printed information may be needed before the program begins (role-play briefings, or program-handling instructions, for example).

If a program provides for a print-out, the documentation should make clear what types of printer can be used, and there should if necessary be an installation program so that the user can configure the program permanently for the printer commonly in use. The alternative is to set up the printer every time the program is used, which means that every user must know the configuration procedure.

There should also be provision for using the program when the computer is not linked to a printer, or the printer is not switched on. A "user-friendly" program will ask the user whether a printer is attached to the computer, and then offer a reminder if the printer is not switched on.

If the program tacitly assumes that the user has access to a printer, and that the printer is switched on, then it may simply refuse to proceed when it gets to a point where it feels it should be sending something to the printer, and often there will be no indication to the user of why everything has suddenly come to a halt. If the printer can be switched on at this point, the program will continue, but if there is no printer to switch on, then the only way out is to abandon the program.

Record-keeping

Computers have no difficulty whatsoever in keeping tabs on their human users. One horror story tells of a university computer which required students to "log on" by name before starting work, and then directed them to use a particular program or course which, in the computer's estimation, was right for them. The computer's judgement was based initially on information fed to it by a course tutor or the college authorities, and then developed according to the computer's "opinion" of each student's progress. The frightening aspect was that the computer did not allow the student any say in the matter. The story goes that a student whose performance was not up to scratch was told by the computer to go away and study a particular textbook for two weeks before coming back to try the same program again. After one week, the student felt that she had made considerable progress with the work which had been set, and logged on to the computer for another attempt. The computer responded with a message which pointed out that it had stipulated *two* weeks' revision, and asked the student not to come back until the full two weeks had expired.

Apocryphal stories like this are not too far short of the truth. Programs for mainframe computers could easily record the work of hundreds or even thousands of students in this way, and even tiny micros can log the attempts made by a particular student within a particular program, so that the teacher can later check up on that student's progress.

Few teachers would approve of keeping records on a student for the purpose of forcing that student to progress through a particular course of study at a pace chosen by the computer, but some may feel that other kinds of record-keeping are valid.

Clearly there are ethical considerations to be taken into account when deciding whether or not to use record-keeping facilities in computer programs. The following are examples of instances in which records of the use of a program may be kept:

- Programs under development need to be validated (to make sure they are technically sound) and evaluated (to make sure they are pedagogically sound). A programmer may include a record-keeping sequence in a program which is under trial, and remove it from the final version. The users are told during trialling that they are "guinea-pigs" and their use of the program will be logged, but that no record will be kept of their names.
- Programs which are too long to complete in one session may store details of the user's progress to date, so that the user may return later and continue the work. The sole purpose of keeping these records is to facilitate ease of use of the program, and the records are not meant for checking by a third party. They would not, in any case, be held in a form which a third party could check.
- A program which follows different routes according to progress (that is, it provides reinforcement routines for students who seem to be having problems) will keep internal records of the user's answers, but these will be destroyed as soon as the user abandons the program.
- Some programs provide an internal routine to keep records of the user's name and responses. A separate program is provided for the teacher to open up the student's file at a later stage and check what work has been done. This kind of record is usually kept "behind the scenes", without the knowledge of the student, unless the teacher chooses to explain what is being done. Many educational programs for primary and secondary children contain this kind of routine.

With the exception of the last, all the examples of record-keeping described above are legitimate in that they constitute no attack on the user's privacy, and make no attempt to record information about students without their knowledge.

The last example is rather different, in that it is possible in such programs for the teacher to "spy" on students who thought that their work was private. It may be argued that the use of the emotive word "spy" is unjustified, since the teacher is merely monitoring the student's progress from the best of motives. There might possibly be a case for record-keeping which was overt, in which the student knew in advance that responses would be logged and later examined by the teacher, especially if teacher and student examined the student's file together. Nevertheless, knowing that this had happened in one instance could undermine the student's confidence in the privacy of other programs.

Adult students usually study voluntarily. Many adult students have had unfortunate experiences of the education system earlier in life. These types of student in particular resent unwarranted intrusions into their privacy.

Moreover, data protection is at present an unresolved issue, and increasing numbers of students are engaged in studies which involve continuous assessment. People have a right to know what information is being kept about them, and what is being done with it, and this is no less true in the field of education than in any other area of life.

On balance, there is little in favour of internal record-keeping. The benefits are heavily outweighed by the potential mistrust of the computer which many students would feel if they suspected that "Big Brother" was breathing down their necks, so to speak. Programs which do store records should announce this fact to potential users before they start work, and perhaps offer the user the option of keeping records or not, so that it is the user's own decision whether or not to allow the teacher to check the work. Far better, though, that users should be able to feel confident of absolute privacy in their use of the computer, through knowing that *no* programs could be used in this way.

Software under student-control

The keeping of "secret" records is one way of destroying student's confidence in the computer. Another way is to allow the computer to take over the proceedings, instead of leaving control in the hands of the users.

Very few programs — thank goodness! — present page after page of information in book-like form. *All* CALL programs demand more student-participation than the average book — even a textbook — if only because at frequent intervals the program stops and demands a response from users, who cannot ignore the difficult bits, or peep at the correct answer and pretend to themselves that they would have got it right anyway. Good CALL programs not only require the attention of users more than books do, but also give a freedom of choice and decision which cannot possibly be matched by a book, however good.

Control over length of study

Programs should, for example, provide a variety of escape-routes, so that the activity is as far as possible under the user's control, which helps to prevent the feeling of being lost in a program whose dimensions are an unknown quantity. This means that, unlike old-fashioned programmed learning, CALL programs need to be flexible, and allow the user to control when to start and finish, how long to spend on an activity, and how fast to work. If the chosen activity involves a set number of questions, the user should be told *before* starting work how many questions are involved, and then kept informed of how many questions have been answered as the program progresses. Better still is the program which allows the student to opt out at any point, so that the user is at all times controlling the machine, never the other way round.

Control over speed of study

One aspect of control which is especially important in CALL is control over the rate of progress. Some programs operate on "timed pauses", which means that the programmer decides how fast the program will move, and it is the same speed for each user. In CALL programs, especially, (unless speed of reaction is a critical feature of the program) students must be in control of the screen display, which should only change significantly when they are ready to continue. Continuation should be a conscious choice of the user, in part because students should know that they are controlling the machine, but principally because CALL students of differing abilities will have different needs, and may want to study the screen or write down some of its contents before continuing, or may wish to work faster than the preconceived notion of the programmer allowed for.

A program in which the student is attempting to improve reflex responses by, say, getting through as many examples as possible in one minute, should provide feedback after each response, rather than allowing the user sixty consecutive seconds during which every response may be wrong. In this kind of exercise, the program should "stop the clock" after each choice, to allow as much time as the student needs to examine the correct response or write it down.

If a sequence uses animation techniques, it may be advisable in the first place to run the animation from start to finish at a set speed. In this case, there should be an option to see the sequence again, as often as needed, and subsequent runs may include a "freeze frame" facility, so that students can stop the sequence in mid-action to inspect it at leisure.

Leaving students free to choose

Other ways of putting choice in the hands of the user may include the provision of a variety of speeds of presentation in a speed-reading or observation exercise, asking the user to select the frequency of gaps in a cloze passage, giving an option to see or not to see a phrase or passage before the exercise begins, calling for a decision on whether or not to see clues in the course of a program, or offering varying levels of difficulty.

The Geoffrions, in *Computers and Reading Instruction* (Geoffrion, 1983), point out that

> arcade games provide an excellent model for manipulating difficulty. Most games start very easy to give beginners a taste for success and then gradually increase the challenge until they are difficult for experts . . . Some (students) may be overwhelmed by a program's complexity, whereas others may find it quite unchallenging. Better-designed software gradually increase the challenge or permit students to select their own proficiency-level before commencing.

It is here where authoring systems which allow teachers to enter varying levels of data can score over fixed-data programs, especially if newly-entered data appears on the menu of choices at the start of the program.

Freedom to recap

One way in which books often score over programs is in the ready accessibility of any page. A student reading a text from a book may need to "page backwards" for a variety of reasons, and the same facility should be available on the computer, when appropriate.

A program which involves presentation of material which may be new (and that includes extended reading texts) should allow the user to move backwards as readily as forwards, certainly to the extent of recapping on the previous example and sometimes allowing the user to start the program again. Every part of a reading passage should be available to the user at all times (unless, of course, recall is part of the object of the exercise, and even then a "peeping" facility might avoid frustration). "Paging backwards" should also permit recall of instructions, introductory material, and commands for operating the program, all of which should, of course, be written in plain, unambiguous English, avoiding jargon.

Accessibility of software

Drawing another parallel from print-literacy, a student who wishes to obtain information from a book can find the book through the subject- or author-index in the library, then start at page 1 and read through from beginning to end, or skim through the contents and turn straight to the required subject, or check the index and look up all the references to a particular entry, or leaf through the book, hoping to catch sight of something relevant.

A body of information on a computer can be searched in much the same way, so long as the user has access to a database of details of available software. Imagine a library in which the books were placed on the shelves in no particular order, and for which there was no subject- or author-index. Searching for a book would require reading the spine of each book in turn until the right one turned up. To make the task more difficult, imagine that, to save space, several books had been condensed into each book-cover, with one generalised title written on the spine. Now, every book would have to be taken of the shelf and its contents-page examined in order to track down a particular title. To make things worse, suppose there was no contents-page: now the searcher would be obliged to read a part of each book before establishing whether it was the right one or not. The alternative would be to provide an omniscient librarian, who could either send you straight to the book of your choice, or pick something off the shelf which "might suit you".

This is in a sense exactly what most teachers are doing with computer software. Even the present tiny range of CALL software extends over a considerable number of discs or cassettes. To save space and money, several programs are placed together on one disc or cassette, probably because they fit the space available rather than because they are subject-related, and a single "umbrella" title is written on the label. A hotch-potch of discs or cassettes is places in a box on a shelf, and the potential user tries first one, then another — or relies on the teacher to say "You'll probably like this one!"

Teaching students how to find out what is available should be a basic element of computer literacy. The snag is that in most cases access to computer materials is as haphazard as it would be in the nightmare library described above, and apart from one or two members of staff with a special interest in the micro, the majority of teachers will know little more than the students about what is on the shelf, and where. As the body of CALL software grows, so will the problem. The difference between print-software and micro-software is that, asked by a student for printed information on a particular subject, most teachers would either recall a suitable book-title or exercise from memory or would at least know how to track one down in the library, whereas few teachers would know how to track down a computer program which could provide the same information.

This problem must be solved before the body of software becomes even more unwieldy, and good material goes to waste simply because nobody remembers it is there. If the problem is not solved, the use of the micro will become a haphazard affair depending on the luck of the draw, rather than on the conscious decision of a student or group of students to follow a particular line of enquiry, and control and use of materials will remain in the hands of one omniscient teacher, instead of in the hands of the student, where it rightly belongs. If the computer is to be anything more than a toy or a white elephant, software must be made accessible to potential users. Just as we expect to have a librarian responsible for organising materials in the library — cataloguing new items, removing or repairing damaged stock, and so on — we should have a person responsible for organising computer materials. Just as the librarian updates catalogues of printed matter in the library, so should catalogues of computer software be prepared and then updated with each new acquisition. It makes sense to have one person with this responsibility — for the same reason that we do not individually add our own choice of books to the library stock and then make new catalogue-cards or microfiche entries for them.

Catalogues of computer materials can most easily be held on the computer, in a database. The database can be a kind of electronic card-index, cross-referenced so that the student can use it as a combined contents-page and index. This would enable a search to be made for general categories (what simulations are there? what is there in the way of adventure games?), or for specific subject areas (what programs are there on phrasal verbs? is there anything about cars? what can I use to help with writing letters?), and these searches could be made using the same techniques on the same index.

Pioneers in CALL will probably be obliged to buy an all-purpose database and set it up to suit their own methods and range of software. In one way, this is an advantage, as it allows the choice of subjects and their organisation to arise naturally from the needs of the educational community. But no doubt before long programmers for various machines will come up with purpose-built databases for cataloguing CALL materials. The important thing is to ensure that, one way or another, both students and teachers have access to details of what programs there are, what they do and where to find them.

The job of teaching students how to question the database to find what they need should not be a burden that falls on one pair of shoulders. All teachers know how to "question" the library catalogue. In the near future, all teachers will *need* to know how to question the software database, so that they can pass the information on to their students.

Drawing conclusions

The fact that computer programs are not like books does not make them any better or any worse than books — only different. Although new skills are required of both teachers and students for CALL compared with those needed for more traditional learning methods, they are skills well worth acquiring.

Once the materials are catalogued and made easily accessible, the biggest problem in acquiring the new skills is that of overcoming what is, for some people, a deep-rooted fear of technology. Any computer program worth its salt should be capable of being mistreated and coming through unscathed, so it should be possible to demonstrate to newcomers that any amount of key-pressing will not harm the machine or the program, and that errors produce intelligible instructions for proceeding with the program. Group work could reduce the fear of "doing the wrong thing", because responses will be a joint effort, and timid students can watch others doing the key-pressing at first. Once confident that they are not likely to do any damage, students should find that on-screen menus and instructions carry them safely through any program without hitches.

The choice of programs which will be hitch-proof is fraught with difficulties of its own. As with textbooks and audio software, much of what looks marvellous in the shop will prove to have drawbacks in practice. In addition to the guidelines already mentioned, the following are worth bearing in mind:

☐ will the software serve a useful purpose for present and future generations of students, or will it be a nine-days' wonder?

☐ could that purpose be served better or more cheaply by other methods or media?

☐ can you find any other users of the software whose opinions you could canvass?

Finally, be prepared to complain loudly and, if necessary, publicly, if you are not satisfied with the quality of any piece of software you have bought. This may mean that you will ultimately receive an adequately "debugged" product, with the grateful thanks of the programmer. At the very least, you will be helping to protect other purchasers from the same fate as yours.

Chapter 4

Hardware — today and tomorrow

Computer programs for language work (some already available, some under development, and some no more than ideas) have been discussed at length, but so far very little consideration has been given to the choice of hardware on which to run the programs. This has been a deliberate policy, so as to avoid tempting prospective users to put the hardware cart before the software horse.

There are two questions to answer:

☐ what software can I use?

☐ what machine shall I buy?

— and it is important to tackle them in that order.

All too often, the questions are reversed, and computer-buyers find themselves buying a machine with all the latest features, only to find that it will not run the software they want to use. The rule should be, if possible, to examine the range of software available for your subject-area, see which machine(s) it is written for, and only then choose the hardware.

This ideal cannot always be met, however. In many cases in Further Education, a college will already have machines which are being used in other areas of work, and pressure will be put on language-teachers to use the existing equipment, even if there is no suitable software for language applications, and even though the equipment will probably be in an unsuitable environment for language study. In Adult Education the situation may be even more difficult, since the choice of equipment may be limited to whatever happens to have been chosen for use in the host school or college, which may or may not be made available for the use of the AEI.

Even if new equipment is to be bought, there could be problems. The Kennings, in *An Introduction to Computer-Assisted Language-Teaching* (Kenning, 1983), write:

> There may also be a choice between several systems and the teacher may have to decide which is most likely to be suitable, and, although he or she will probably have less say in the purchase of new machines than colleagues in science departments, it may be possible to influence decisions.

The authors make an unquestioned assumption that computers really belong to science, and that language work must be a poor relation, even to the extent of having to make do with unsuitable equipment bought primarily for other applications.

All these attitudes must be fought strongly, if computing for language students is to have a future. The following factors are vital:

☐ according to the FEU recommendations, *all* students have a right to instruction in computer literacy, using materials relevant to their course of study;

☐ setting aside considerations of "pure" computer literacy and/or linked skills, language students need purpose-written language programs, and access to the appropriate equipment for running those programs;

☐ language teachers should not be obliged to bow to the "superior" claims of other departments when it comes to specifying equipment for their own applications;

☐ language departments have as much right to computer equipment as any other department, and their students have as much right to computer education as any other students;

☐ the peculiar needs of language study mean that the siting of computers primarily intended for scientific applications will probably be inappropriate; a smaller amount of equipment sited in a suitable environment would probably be more valuable for language-learners.

Choosing computer hardware

Assuming freedom of choice, there are some general principles which may be applied to the purchase of computer hardware, bearing in mind the following factors:

☐ whatever equipment you choose will be out of date almost before you sign the order-form;

☐ there will always be something bigger, better and cheaper on the way, if only you had waited for it;

☐ some computer manufacturers are here today and gone tomorrow, so it seems wise to choose one who is likely to survive; the ones with government backing have a reasonable chance of survival;

☐ the need for a reliable system, supported by a good range of software, and preferably one which is compatible with other machines from that manufacturer's range — if you are about to invest in software for the XYZ, it is comforting to know that much of it will also run on Son of XYZ; should you wish to buy another machine at a later stage.

Recent years have seen a massive escalation of interest in microcomputing, and a corresponding increase in the availability of new makes and models of micro. It is true that there is always a new, improved machine just about to be marketed, and if you wait for the best you will never buy. The other side of the coin is that if you jump on an early bandwagon, you may be stuck on it when everybody else has jumped on a different one. This is much more the case with computer hardware than, say, with audio equipment; today's audio cassettes will run on tomorrow's cassette-recorder, whatever the make, but computer software is not going to run on just any computer, new and wonderful though it may be.

Market forces, the LEAs and CALL

Two makes of microcomputer — BBC and Research Machines (RML) — dominate the British educational market as we enter the second half of the 80s. So fast is the microcomputer world changing, however, that this could be merely a temporary monopoly. The following comparisons are therefore intended only as a guide to some of the forces at work when it comes to choosing an educational micro, rather than a hint to the reader that these are the makes/models to go for.

The BBC has price in its favour. It is comparatively cheap to set up a basic BBC system, which can then be upgraded by degrees. This has encouraged many schools and colleges to opt for the BBC, even though the 480Z is in many ways superior to it. The Manager's Guide on *Computer Literacy* (Lloyd, Taylor & West, 1984a) advises:

> The BBC microcomputer has been identified as a possible best buy for Computer Literacy courses, in view of its cost effectiveness, robustness, versatility and software and hardware support.

The British Council is setting up pilot projects in six centres overseas, based on the BBC micro, and is developing its own language-learning software.

In terms of spending power, the educational market is small and, generally speaking, publishers will not invest in it unless they can also drum up a home-software market. The home-market for language-learning materials being negligible, the publishers will be looking elsewhere for sales. CILT representatives make the point that, with the major potential market in EFL materials being overseas, publishing houses will produce software to meet overseas demands — which means, in effect, software for the BBC, IBM, Apple, and Spectrum. Not for the RML models.

Yet a hardware comparison of the BBC Model B with Research Machines 480Z must favour the 480Z in almost all but price. However, cost is so effective a spur that the BBC sells in much greater numbers than the 480Z, consequently the profit potential in software production is greater, and thus more software is written for it, which in turn influences even more people to buy it.

Two strong arguments in defence of the 480Z are

☐ many educational establishments are now installing "networks" — that is, a number of computers linked together to form a single system — instead of individual, stand-alone machines, and the 480Z network not only compares very well in cost with the BBC network, but is also widely considered to be a better, more flexible system;

☐ the BBC uses its own very idiosyncratic operating system, and (in its basic form) can run only programs specially written for it; the 480Z, on the other hand, uses an international operating system — CP/M — and therefore has access to a much wider range of languages and software developed originally for other machines.

At present, there is little language-learning software for either BBC or RML machines, but the private sector is developing programs for the BBC, and there is also a vast quantity of BBC software from other disciplines which may be borrowed for language-learning purposes — even if much of it is of poor quality in educational terms. The memory limitations of the standard BBC machine, however, will undoubtedly mean that many of the desirable features of future CALL programs (such as those described in Chapter 2) will be impossible to achieve on this machine.

Whatever present-day machine is chosen, imminent technological developments are likely to make it obsolete within a comparatively short time, and it is therefore important that programs already developed should translate easily to the new generation of machines.

If LEAs were to support the development of language-learning materials on an adequate scale to satisfy the needs of EFL, ESL and Adult Literacy students, the spin-offs for computer-aided learning across the curriculum would be considerable. Many of the programs developed specifically for language-learners would in fact find applications in other spheres of education, and at all levels; and there would also be far-reaching benefits from drawing language teachers, who are specialists in communications, into the team-development of CAL materials for other subjects.

Choosing hardware: financial considerations

Setting aside the availability of software, which must be the most important criterion in the choice of hardware, finance must also be taken into account. The question of whether two cassette-based BBCs is a better choice than one disc-based 480Z is academic, if you are an AEI with no money for computers at all. The following comments drawn from the London-wide survey of computer provisions for ESL, EFL and Literacy (see Chapter 5) were typical:

> In adult education we work to a tight budget, which has not allowed us to use computers as yet.
>
> This is something we hope to do in the future but no money has yet been allocated to this area of development.
>
> We are trying to get funding for one.
>
> Refused permission to install computers.

The Computer Literacy requirements set out by the FEU must surely lead to money being made available to *all* sectors of education, in order to allow the recommendations to be put into practice. And since the aim of the recommendations is to promote computer literacy as an essential life skill, it follows that there must be a provision of hardware and software of suitable quality, and on a suitable scale, to enable that aim to be achieved. A reader's letter in *Educational Computing*, November 1983, asked:

> How much is it actually costing LEAs to support Acorn's near monopoly

> market position? Would twice as many computers not give a much better educational return to those LEAs than the present situation does?

In other words, the BBC — produced by Acorn — is too expensive, and twice as many Sinclair Spectrums would do a better educational job. Another letter in the same paper, in May 1984, presented the other side of the coin:

> There is also the problem that our students often seek employment with small businesses and I have yet to find a small business that uses a BBC micro for either data or word processing.

In other words, older students need access to more professional (and more expensive) equipment, and one IBM would be more appropriate than three BBCs.

Compatibility and compromise

These arguments are going on throughout the education service, as educational establishments attempt to pull themselves up by their bootstraps into the microcomputer age, usually finding that their bootstraps are mere shoestrings!

The appropriate compromise between quantity, quality and price will differ from one application to another, and will vary in direct relationship to the quantity and quality of software available. There is an additional element in hardware purchase for language-learning applications, however, and that is *future* compatibility.

New micros are flooding the market, and "educational" software of good, bad or indifferent quality follows close behind. But while the machines are tending to get better and cheaper, the "educational" software remains mostly bad or indifferent. Or, for purpose-written language-learning applications, largely non-existent.

If your application can be satisfied by off-the-peg business software, almost any machine with a typewriter-pitch keyboard will do. You can buy word-processing packages for most machines, even the very cheapest, and they all have their own versions of financial spreadsheets and databases, so you can buy the latest machine knowing that your needs will be catered for.

But if your application is of a specialist nature, as with language-work, and the software is hard to come by, or you have to write it yourself, what happens when a new micro emerges, which has wonderful new features, but is totally incompatible with all your specialist software? What happens if a boffin in the educational hierarchy suddenly dumps a new and incompatible system on you — all in the name of progress — and takes away the one that actually runs your programs?

All these considerations — of software availability, hardware quality, compatibility, and finance — mean that whatever choice is made will be a long-term choice, and so must be made with care.

Choosing hardware: initial decisions

What are the questions which need to be answered before this choice can be made? Using the above guidelines, the choice of equipment to use for language-based applications will already have been narrowed down considerably, unless you intend to write all your own software, but there are still options to consider once the initial choice of the computer model has been made. The following description is not intended to be a complete guide to computer purchase, merely to offer a few starting-points for first-computer buyers. There are many books and magazines which explain the parts of a computer and how they all work, and these can be referred to if more detail is required.

You will need, in addition to the computer itself, a means of storing programs (either a disc-drive or a cassette-player), a screen (called a VDU — visual display unit — or monitor) for viewing the programs, and possibly a printer. Some of the basic options for consideration are:

☐ the memory-size of the computer;
☐ whether to have disc-based or cassette-based program-storage;
☐ whether to opt for stand-alone machines or a network;
☐ whether or not to have a printer;
☐ whether to economise by having mono instead of colour.

Memory

Computer-memory is measured in "kilobytes" or "K". Nowadays, 32K is a small machine, 256K is capacious. The old joke that says whatever memory you buy today, you will need twice as much tomorrow, is not really a joke, so the sensible thing is to buy as much memory as you can afford. Many of today's programs for language-learning are small, but future applications will undoubtedly be more sophisticated and require more memory to run in.

Also, if you want to run a reasonable word-processor, which will allow you to make documents of substantial size (say, for multi-page class handouts, or for administration purposes), make sure that the model of computer that you choose has enough memory to run the word-processor you want, and to allow you to create documents of the maximum size you will need.

Program storage

The choice between a disc-based or cassette-based machine is really no choice at all. The only consideration which should prevent the purchase of a disc-based machine is a financial one, as disc-drives can cost twenty times as much as cassette-players. If financial constraints force the choice of a cassette, be sure that the software you want is available on cassette and not only on disc, and also be prepared to upgrade your machine to discs as soon as you can afford it. And, in the same way that one disc-drive is better than a cassette-recorder, two disc-drives are better than one.

The chief advantage of disc over cassette is that of speed — a program which can take frustrating minutes to load from a cassette can be loaded from a disc in a matter of seconds. And with the disc you can almost guarantee a successful load at the first attempt, whereas cassettes are more prone to error on loading, which means embarking on a repeat performance.

But another very important consideration, especially in language-based programs which may involve a lot of data (vocabulary, stories, phrases, etc.), is the fast access to data files which discs can provide.

Many cassette-stored programs

☐ will include lists of data within the program, which means that the programs will take even longer to load, especially if they contain large quantities of text;

☐ will be limited (by the computer's memory size) in the amount of data which can be accessed;

☐ will take some technical knowledge to modify (always supposing that the program allows "breaking and entering" for the purpose of modification, which most commercial programs do not);

☐ will require you to save different versions, if you can modify the program and then wish to provide a variety of sets of data.

The provision of a good authoring facility on a cassette-stored program is the exception rather than the rule, and so most cassette-stored programs suffer from the above drawbacks.

Disc-based programs, on the other hand,

☐ need not store data within the program, so the program loads even faster, and then quickly "reads in" data stored on separate files on the disc. This means that vast amounts of data can be stored, limited only by the disc-storage space, which is much greater than the memory of the computer itself;

☐ need only feed into the memory the set of data required for one particular "run", the choice of data being made by the user from a menu at the start of the program;

☐ can operate on a large number of sets of data via a single version of the program;

☐ can provide an authoring-system, to allow entry of "customised" data by users without programming experience.

Add to this the difficulty of finding a particular program on a long cassette, compared with the ease of a self-organising disc with its own internal directory, and the advantage of discs are clear.

Stand-alone machines or networks

The choice between stand-alone machines and a network is less clearcut. If it is intended to limit the number of micros to three or fewer, then independent, stand-alone machines would be the obvious choice, or, in the case of the 480Z, a "shared disc" system. If more machines are required, a network becomes a cost-effective option, although it may not be the wisest choice in other respects.

The advantage of a network is that it provides a "chain" of machines, all drawing on one central operating system, one set of disc-drives, and one printer, which is obviously more cost-effective, the more machines you add to the network. The disadvantage is that networking can slow programs down considerably, because a number of users are all drawing on the same equipment. This problem is accentuated when the programs in use require frequent disc-access, which may be the case with, say, word-processors, or adventure-games.

Furthermore, it is vital before opting for an expensive network to ensure that it will do all the things you want it to do. Users who purchased the Level 1 form of BBC network, for example, were dismayed to find that they could not use programs which drew on external files, so that all the disc-data-handling advantages mentioned above were wiped out at a stroke, and programs had to be re-written with internal data statements, as if they were to be run from cassette. This same network also placed a limit on program-size of 12K, which meant that programs either had to work within this tiny constraint (including data, which can take up a substantial amount of memory), or had to be broken down into small portions and "chained" together. At least one educational centre using this network very quickly decided to upgrade all the network machines into stand-alones, by buying supplementary disc-drives; another had to wait impatiently for the more expensive Level 2 network, which did not suffer from the same limitations, to be marketed.

On the other hand, although networks can be slightly more difficult to use than stand-alone machines, they have some important advantages other than cost. An excellent account of "The Network in Action" appeared in *Educational Computing* (Fluck, 1984), giving an overview of what a well-run network is capable of. The system described is at Stantonbury Campus, covering several sites and providing facilities for staff and administrators as well as students, but some of the ideas would be applicable on a smaller scale. One of the most imaginative applications in language terms is the use of an "in-house magazine":

> This latter program is loaded into any BBC computer station that is switched on while connected to the Econet. It hunts for a viewdata frame called PAGE 100 and its successors which form a loop of up to about 20 frames. Once loaded, these frames are displayed for 15 seconds each in continuous rotation . . . Thus a terminal which is switched on but undisturbed or unused, acts as a noticeboard and introduction . . .
>
> However, every frame in this cycle also carried the message PRESS

> ANY KEY FOR FACILITIES. If a person wishes to use some of the software held on the system, a key press will bring up a menu page . . .

The magazine contains advertisements for school activities, "tasters" of a selection of the programs on offer via the network, messages for individuals or classes, and so on. The Bell School of Languages in Saffron Waldon, although its network is on a much smaller scale than that of Stantonbury Campus, runs a similar viewdata mazagine, which they have called "BELLTEXT". Here the "magazine" pages are prepared and entered by interested students, and one of the monitors is placed so as to be visible to students passing along the busy corridor outside the computer room. Because the network is always switched on for the magazine display, entry into the program of the user's choice via a few key presses (carried out using screen menus) is very simple.

Portability

An undoubted advantage of the network is that it is bolted to the bench and thus more secure. The corollary is that it cannot be shipped from room to room according to need — although this can be a mixed blessing, as explained in an *ILEA Computers in Primary Schools* Newsletter (Laws, 1984):

> We are still in the unpleasant situation of having to unload our computer trolley and lock away everything in a safe/strong box. This coupled with the difficulty of manoeuvring the trolley through narrow doorways is no incentive to anyone wishing to use the equipment.

Add a few flights of stairs to this basic problem, and the real drawbacks of the government's objective of "one computer per school" become evident!

Printers

A printer is an essential addition to the hardware

☐ if word-processing facilities are to be provided;

☐ if programs provide a print-out for students to take away (this is usually optional — that is, the program can be used with or without a printer attached — but a printout is a desirable feature);

☐ if teachers use, say, a Wordsquare program to prepare a class-set of handouts;

☐ if home-made programs are to be written, as these will need to be "listed" on paper;

☐ if the computer is to be used for administrative purposes, as well as for student use.

For a small number of machines, one printer may well be sufficient, and unless it is to be used to produce high-quality work (such as camera-ready copy for publication, or correspondence), a cheap dot-matrix machine will be adequate. A dot-matrix printer is fairly fast, fairly quiet, fairly cheap, and produces fairly good copy. A daisy-wheel printer is very noisy, very slow, very expensive (though getting cheaper), but produces excellent electric-typewriter-quality copy. On balance, some kind of printer is advisable, but for every-day use it need not be a top-of-the-range model.

Monitors

Finally, having provided the micro itself, a means of storage for your programs, and a way of printing out "hard copy", you will need a screen to view the programs. The options here are:

☐ an ordinary TV or a dedicated computer monitor?
☐ monochrome or colour?
☐ high, medium or low resolution?

A monitor is preferable to an ordinary TV set, as the screen is capable of showing more detail and produces a steadier picture. The fact that a monitor does not require a television licence, since it cannot receive external broadcasts, may also be a significant factor in some cases. However, if a television set is already available, and if the chosen computer can operate with a standard TV, then using an existing set may offer a cheap way into computing.

A monochrome monitor may be only half the price of a similar-sized colour monitor, and this could also be a way of making economies in the initial outlay. Although programs look more attractive in colour, there is no reason in most cases why they should not be viewed in "black and white" (or green or amber, depending on the monitor you choose), at less than half the price. But remember that some programs *depend* on colour for their successful operation, and these will be unusable with a monochrome monitor.

Most computers now offer colour as standard, but if your chosen make of computer allows a choice of mono or colour versions, economies can be made not only in the price of the monitor, but also in the micro itself. When sufficient money is available, the micro can then be upgraded by the purchase of a "colour board", at the same time as a colour monitor is bought.

The choice of "resolution" — how much detail the screen can show — depends on how much resolution your micro is capable of, as well as how much money you can afford. An ordinary television screen may possibly give a usable picture — with the proviso that some micros will not operate with an ordinary TV at all — but the improvement offered by a monitor is considerable. Most micros now offer "high resolution graphics" as a matter or course, and it is a pity to degrade a program by running it on a TV or monitor which cannot show it to its best advantage. On the other hand, if the monitor is capable of much higher resolution than the computer, then you are throwing away a considerable amount of money on a feature which your micro cannot use.

If you intend to use a word-processor regularly, and the program you choose has an 80-column screen display, the writing on the screen will be small, and a good resolution will be required to display it in a way that will not tire the eyes. A high-resolution monochrome VDU may be more suitable for this purpose than a medium-resolution colour monitor, and so it is advisable if possible to "try before you buy".

General principles

As a general rule, it makes more economic sense to buy a basic, no-frills version of a good computer, which can be built up in stages with extra money, disc-drives, colour and so on, than to buy a cheaper model which has all the features built-in, but which will probably lack durability, and may not run the programs you want to use.

Future Developments in CALL hardware

There is not going to be an explosion of interest in CALL outside the immediate circle of those involved with language-learning techniques. No technological whizz-kids will suddenly see the light, and devote their lives to applying the latest developments in microcomputer technology to improving the lot of the language-learner. Nevertheless, CALL can thrive by taking advantage of the innovations which are undoubtedly going to be developed for other, more profitable applications of computing.

The QWERTY keyboard

One of the developments of most interest in language-based applications is the growth in alternative methods of communicating with the computer. A keyboard is not the most efficient way of putting information into a computer. The QWERTY keyboard (so called from the order of its first six alphabetic keys) is a hangover from the days of early mechanical typewriters, when the keys had to be organised in such a way as to prevent the typist from typing too fast, since the primitive system of mechanically-operated rods could not keep up with the potential speed of the human fingers. The old-fashioned keyboard arrangement is now so firmly entrenched that attempts to replace it with something more satisfactory have met with failure. The millions of typists the world over who have trained on the QWERTY keyboard (allowing for slight variations to the order of the keys in non-English-speaking countries) have naturally resisted change, and this leads to more generations learning QWERTY, thus digging the system even deeper into our culture.

The problem with the QWERTY keyboard is not simply the order of the keys; despite their illogical arrangement, a good typist can achieve quite phenomenal speeds. The real drawback is that the QWERTY keyboard needs to be *learned*, and a non-typist cannot expect to find the right keys easily, even if only a small selection of letters is required for a particular purpose. If an extended input is required, the non-typist is not merely at a slight disadvantage, but is virtually cut off from ready communication with the computer. Watching a non-typist entering the word "YES" via the keyboard can be a painful experience.

The use of an ordinary typewriter differs from the use of a computer in one important respect: the typewriter has only one use — to produce printed documents — and in order to take full advantage of the machine, the user expects to have to learn the keyboard. The computer, on the other hand, has an unlimited variety of uses, many of which are only indirectly involved with the keyboard — in fact, the keyboard is more often used merely as a means to an end, the end being to communicate with the computer, not to produce printed words at all. It is obviously impractical to expect all potential computer-users to become familiar with the QWERTY keyboard, especially when their use of the computer may be very limited, and their time even more so; and when their object in using the computer may have nothing whatsoever to do with the keyboard. Another problem is that, while concentrating on the keyboard, the user may be missing important happenings on the screen.

This situation has led to the development of a wide range of alternative ways of entering information into the computer, some of which lend themselves to language-activities more than others.

Variations on the keyboard theme

Modifications to the keyboard itself may be as simple as attaching coloured stickers to certain commonly-used keys in order to draw attention to them. This is not a very satisfactory solution, however, as different programs use different keys to perform the same function — for example, on the 380Z, WORDSTAR uses the E, X, S and D keys to move up, down, left and right on the screen, but the simulation SLICK! uses I, M, J and K for those same screen movements, and many programs use 1 and 0 or < and > to move left and right. With half a dozen programs in general use, the keys with stickers would soon outnumber the ones without.

Many computer keyboards now have four "arrow keys", whose sole function is to move the cursor around the screen, and this is a useful development. Another variation in the standard keyboard is the provision of "function keys", a set of keys above the main QWERTY arrangement, which can be programmed to perform quite complex tasks in response to a single key-press; however, this still does not solve the problem of having to learn which keys to what, especially when every program uses the function keys to carry out tasks specific to that program, and key F1 might delete a block of text in a word-processor, select a text in a cloze program, fire a lazer at a space-rocket in an arcade game, or load a program from a menu, depending on which disc you decide to use. The more professionallyproduced programs supply an overlay for the function keys, to act as a reminder of their roles.

Keyboard alternatives

The other possibilities involve attaching an extra piece of hardware to the computer, either as an alternative to the keyboard, or more commonly as an addition to it. Joysticks and track-balls have been introduced as a means of playing computer games involving moving symbols round the screen. The fact that they are usually used to "fly a space-ship" and "fire a laser" need not prevent their more sober use in, say, moving an arrow up and down through a menu, and pressing the "fire-button" to select an option. It is unlikely, however, that either joysticks or track-balls will be used extensively to circumvent the keyboard for adult language students, as they are suitable only for screen-movement and simple option-selection, and therefore do not help with the more serious problem of typing at the keyboard.

The Mouse

The "mouse" is merely a grown-up, businessman's version of the joystick; it is a wedge of plastic (connected to the computer by a cable — the tail of the mouse) which may be moved around on a flat surface; the computer senses the direction and distance of the movement, and responds by moving the cursor appropriately on the screen. The mouse is, like a joystick, not an alternative to the keyboard, but a supplement to it. Mice may be quite sophisticated in the functions they can perform, and are more sensitive to delicate movement than joysticks, but their disadvantage is that they require quite a large flat surface on which to operate, and that is not always readily available in a classroom.

The fashionable way of selecting options with the mouse is using it to point an arrow on the screen at a picture (called an "icon") of the function you want. Saving a piece of text on the word-processor would be done by pointing at the tiny icon of a filing-cabinet, erasing a text would involve selecting an icon of a wastepaper basket, and so on. The idea is that those new to computers can find their way through a program as simply as possible, which can mean using virtually no language-based instructions at all. And that is another disadvantage of mice for language-learning purposes.

The touch-sensitive screen

A touch-sensitive screen needs no extra hardware add-ons, but it does need a special monitor, which is an expensive way of allowing computer users to make simple choices. Like the other methods mentioned so far, it is a supplement, not a real alternative to the keyboard. The user simply works through screen menus (which may be text-based, or — as with the mouse — icon-based) by touching the appropriate part of the screen, instead of worrying about which key to press on the keyboard. But when it comes to entering facts and figures — and text — the keyboard is still the chief means of communication with the computer.

The light-pen

Perhaps a better candidate for educational computing is the light-pen, which is the poor man's touch-sensitive screen. This can have its uses, as, for example, in the writing program which allows the user to draw directly on to the screen and get instant feedback about the formation of the letters of the alphabet. But even allowing for one or two useful applications like this, the light-pen is more suited to budding Michelangelos than to adult language students, and still does not attempt to deal with text input.

The concept keyboard

The concept keyboard, a touch-sensitive pad, is the chameleon of alternative inputs. It can be whatever the programmer wants it to be. In one program, it might offer simple yes/no and multiple-choice alternatives; in another, it could provide a series of pictures to match with their screen descriptions; a third program may use it as an electronic blackboard, allowing the user to write or draw on the screen in response to instructions; yet another program could be a word-processor, with half the concept keyboard laid out with editing and formatting options, the other half offering the alphabet in its traditional A-to-Z order; in some programs it may be an alternative to the traditional keyboard, in others the user may have a choice of which to use. And all this may be accomplished with a single concept keyboard, with an overlay corresponding to whatever program is in use.

By allowing a rearrangement of the QWERTY keyboard into something more recognisable as an alphabet, the concept keyboard is a potentially valuable means of simple text-entry for non-typists. Like the keyboard of the Sinclair ZX81 computer, it is certainly not ideal for the touch-typist, but for any user not familiar with QWERTY it would open up the possibility of using writing-based programs, and in particular word-processing. For some users, it may even provide an incentive to go on to learn typing skills on the standard keyboard.

The one thing that the concept keyboard does not have at the moment is suitable software to run on it. Its main applications so far have been in banking (what is the Cash Dispenser but a concept keyboard dedicated to providing the customer with fast money?), in some shop-tills (especially fast-food restaurants), and in primary education. A body of suitable language-learning software which offered an option of using a concept keyboard as a primary input could open up the world of CALL to the non-typist and the keyboard-shy, and there would also be enormous benefits in literacy applications.

The Microwriter

The Microwriter has been on the market for some years now without making the impact it deserves, mainly because of its high price. In its latest version, the Quinkey, it is both affordable and usable, and its operation can be learned quickly, especially by those unused to a standard keyboard who have no traditional keyboard skills to un-learn. About the size of a thick paperback, the Quinkey — surprisingly, in view of its name — has six keys, which are used in various combinations to produce text. The combinations are quickly learned, users claiming familiarity with most of the letters and punctuation in times varying from half an hour to two hours (the record stands at ten minutes). It is operated by the right hand, and since there is nothing to be gained from looking at the keys, users become genuine "touch-typists", which leaves the eyes free for whatever is happening on the screen. The Quinkey can be used by a single operator for text entry to a word processor, or up to four Quinkeys can be "hooked" to one micro, to allow interactive programs to be used — each user having a personal "keyboard". The standard keyboard also remains available for use when the Quinkey is linked up.

The facility of using four Quinkeys with one micro is useful, though not essential. It simply means that groups operate as equals, rather than having one take the role of key-presser on the standard keyboard. This might actually be detrimental in language-learning situations, however, in which giving instructions to the key-presser is part of the learning experience. As a text-entry device, the Quinkey has to be learned, although it can be learned much faster than a standard keyboard; its chief advantage is that it can be used for touch-typing by people who cannot touch-type on a standard keyboard, so that they can concentrate on the screen instead of their fingers. The concept keyboard, on the other hand, does not have to be learned (except insofar as it is necessary to know the alphabet), and the shapes of the letters are printed on the keyboard, which are both points in its favour; on the negative side, the concept keyboard can never be used for touch-typing, so it will mean a constant division of attention between the input on the keyboard and the output on the screen.

Taking all the pros and cons into account, it seems as though the concept keyboard would be invaluable to "casual" users who have neither time nor inclination to learn a keyboard, and to students who need extensive help to improve their language or literacy skills, while the Quinkey approach might suit students of average or above average abilities, whose prime interest is in text-entry, but who do not need standard keyboard skills.

But the overriding factor is still software availability. However useful alternative forms of input may seem, unless there are programs to exploit them, they will be no more than expensive toys with extremely limited application in the language-learning process.

Computers and the spoken word

All these alternatives are readily available, and used in a wide variety of ways in education and business, although their application in the field of language-learning is almost non-existent. None of them, however, offers an ideal solution to the non-typist in any sphere of activity.

For that reason, a great deal of time and money is being invested in persuading computers to understand spoken language, and respond in kind. So far the results are not sufficiently advanced to be of use to language-learners, but the time when listening and speaking micros are commonplace may not be too far away.

Speech-synthesisers

Text-to-speech converters can already "read" text aloud, with largely correct pronunciation, and will offer to spell out words which the listener fails to understand. Using allophones as building-bricks, the vocabulary of the converter can be infinite, but the limitations of present-day technology mean that intonation goes by the board, so that, not only does the computer not understand what it is saying, it doesn't even *sound* as if it understands!

A machine like this can be a boon to, for example, a native-English-speaking blind reader, who, with practice, can interpret the "speech"-sounds with reasonable accuracy. Language-learners, on the other hand, would not have at their disposal all the linguistic clues which a native speaker takes for granted, and there would be no intonation patterns to help, either. Nor would a learner be in a position to question the computer's mispronunciations of words which do not fit the rules which it "knows".

The professional text-to-speech converter is an expensive and sophisticated machine, for all its shortcomings. At the other end of the market, cheap speech synthesisers are produced for a wide range of microcomputers, and cheap is exactly how they sound. Until computers can be persuaded to speak less like daleks and more like humans, speech synthesis for language-learning purpose is probably best left alone.

Speech-recognition devices

But there are two sides to a conversation, and if the computer is not ready for normal speech yet, the computer-user is. A computer which would accept input from the human voice would solve all the problems of the keyboard at a stroke. Once again, the developments which are taking place are mainly for business purposes, but when the breakthrough occurs, the comprehending home micro will not be far behind.

When a computer receives information via a keyboard, it responds according to the instructions which are contained in the program in its memory, without any understanding of either the program, the instructions or the information. In fact, even when the user types in text, the computer views the text in terms of a series of numbers, not as alphabetic characters, and prints symbols on the screen according to their numeric values. The user may interpret the screen-symbols as "writing", but the computer certainly does not. In the same way, the "speech recognition" devices which can now be bought quite cheaply for a number of home micros do not "understand" the words they "hear" — they merely convert them into numbers and then react according to instructions contained in a program.

Fortunately for the user, the computer keyboard always responds in exactly the same way every time it is used. On a machine which uses the ASCII code of interpretation (ASCII is the standard code, which most computers understand), if the "A" key is pressed, the computer interprets it as number 65 if the SHIFT key or the CAPS LOCK key is pressed, and number 97 otherwise. There is only one way of pressing the "A" key. Imagine the difficulties if the computer had to interpret a handwritten "A": every person's "A" would be different, and the same person writing a row of "A"s would not write any two of them in precisely the same way.

This is exactly the difficulty, though technically much more complex, which has to be overcome by speech recognition devices. Even the most sophisticated ones can interpret only single words spoken in isolation, with measurable gaps between words. They only recognise the speech of individuals who have "trained" the device by repeating the same word over and over again until the computer has a good overview of the general "shape" of the word as spoken by that individual. Once trained, the computer works at maximum efficiency in quiet surroundings, with no interference from extraneous noise, listening to a "known" speaker, who must speak very carefully and clearly. Even in these carefully controlled circumstances, the computer has no idea of the meaning of the words spoken, or their appearance in written form.

When the object is to persuade a computer to understand *any* speaker, with all the complications of connected speech, possible ambiguities, with syllables omitted or slurred over, possibly idiosyncratic stress or pronunciation, and a regional accent thrown in for good measure, it is easy to see just how much is still to be done before the "understanding computer" is available to all. Nevertheless, an article on speech recognition in the American magazine BYTE (White, 1984) said:

> Cost-effective, voice-controlled typewriters that accept normal speech will almost certainly not be on the market this decade.

The all-important word is "almost". So fast is technology developing, that no-one can say for sure that it will not be possible, by 1994, to talk to your word-processor instead of tapping out words on the keyboard.

Even the task of building a computer which understands the human voice sufficiently to distinguish, say, "Grade A" from "grey day" in a passage of connected speech will require considerable expertise in the field of artificial intelligence. The machine which will listen to students of ten different nationalities reading the same passage of English, and recognise that it is, in fact, the same passage is still far in the future. But it will materialise, eventually.

Computers and Video

In the area of visual techniques, the two which show most promise for the future of language-learning are digitised pictures and interactive video.

Digitised pictures

With the improved resolution provided on the current generation of micros, pictures of astonishing detail can be drawn on even the most modest machines. The advent of programs like BITSTIK, which enable original screen artwork to be prepared relatively easily for incorporation into programs, means that graphic displays need no longer be the result of long and often painful programming, as they used to be; but if the need is for something more photographic in quality, a digitiser may be the answer. For a (comparatively) modest outlay, a digitiser can be used to produce "stills", which may be single frames from a video film (via a video recorder), or (in conjunction with a video camera) single-frame shots of subjects specially set up for the purpose, or of photographs, maps and so on.

The digitised pictures are reproduced as shades of grey, with a "posterised" appearance. They may be saved on disc, and then incorporated into a computer program. However, a much cheaper method in cases where the original is a photograph or map might be to use a paper copy of the original as a concept keyboard overlay, so that touching a portion of the picture would produce the appropriate result on screen — maybe some relevant vocabulary, a description of the contents of that portion of the picture, or a question to answer.

Interactive video

A visual medium of much greater versatility than the digitiser is interactive video. Although interactive video can be implemented on ordinary video cassettes, laser discs are much more suitable for the purpose. A laser disc contains 54,000 film-frames for each half-hour of running time, so it can hold massive amounts of data, and it is very fast to access any given frame, so that stored material can easily be found. The advantages of video discs over video tapes are, in fact, similar to the advantages of conventional computer discs over audio cassettes — but on a very much larger scale.

Furthermore, video discs are not choosy about what is stored on them. They can store computer programs, audio tracks, still pictures and conventional video film, running first one, then another, under the control of a computer program.

There are many ways in which the potential of video discs can be exploited. They can be used as visual databases, which can store enormous numbers of still photographs, and/or clips of film, and search through them on demand — perhaps producing a selection of frames showing domestic interiors painted by Dutch artists between 1640 and 1650 from a database of paintings, or all the examples of Palladian-style architecture in London, along with relevant text if required.

One example of this application is David Attenborough's *Videobook of British Garden Birds,* which contains the equivalent of a substantial book of text, describing in detail the birds whose pictures can be called up from the database as and when required by the user. If the stills need a spoken commentary, techniques of audio compression enable several seconds of speech to be "attached" to a single frame.

Video versatility

The chief value of the video disc is its ability to combine a variety of media to suit the needs of the user. The Council for Educational Technology's working paper on *Interactive Video: Implications for Education and Training* (Duke, 1983) starts by pinpointing some of the drawbacks of television, which are strongly accentuated when the television material is primarily educational, and/or the viewers are not native speakers of whatever language is being used in the program:

> There is no opportunity for the viewer to pause or reflect, or to take notes while viewing, without losing touch with the thread of the argument, and since pace, level, format, structure and language have been chosen by the producer to suit the perceived average viewer, the appropriateness of any particular programme for any particular student is more a matter of chance than of design . . . there is a tendency towards passive acceptance as the primary learner response.

Interactive video can overcome all these restraints. It is possible to provide menus to allow the user to move directly to the required film sequence, instead of being obliged to use the hit-and-miss search methods of domestic video machines. Text can be superimposed on the film or stills to give flexible sub-titling — the text may vary according to the level chosen by the user at the start of the session. Alternatively, extended pieces of text can be inserted on separate frames at suitable points in the film, or explanatory graphics may be superimposed on part of the film — say, for training purposes, when a video film showing a piece of equipment may be overlaid with a graphic representation of its operation.

Student-video interaction

The user can be required to contribute to the progress of the program, by answering multiple-choice questions or keying in responses of various kinds. Answers can be textually analysed, and program-branching made dependent on the user's responses. The value of this in job-training is evident, but it could also have applications for language-learning, where the ability to make use of the student's voice to cue the disc would also be advantageous. Simple interactive video would allow branching at specific points according to the analysis of the student's response; a more sophisticated video allows the student to break in and divert the course of the programme at will.

John Duke describes one of the systems used in creating interactive video as follows:

> (The WISE system) operates on the 'menu' principle, whereby the author is presented with a range of choices to define the pattern that the lesson material should take and, having made a selection, the system takes charge of the programming detail, seeking from the author just the necessary data for the displays, and criteria for judgement and routing. WISE allows decisions on branching to be made by the author (predictive), the student (free interaction), or by the system (conditional). Student responses are monitored and assessed against different sets of criteria, such as multiple-choice, selection, spelling, word-order, correctness, keyword inclusion, synonym matching, arithmetic accuracy, number of tries, etc . . . A help display can be called up at any time.

The basis of an interactive programme may be video film specifically shot for the purpose, but it could equally well be existing film footage (with due regard for copyright problems), or stills, or a combination of all three, with a mixture of text frames, subtitles, graphics and audio material to suit. The audio material may be recorded in stereo, or as two separate tracks, which opens up the possibility of a single programme with two audio tracks using different levels of English. And all this is possible using little more than a home micro and a video-disc player.

The drawbacks

There are bound to be snags. One of the snags is cost. Even at the cheapest level, it could cost upwards of £30,000 to produce a master-disc for an interactive video programme, although this cost will no doubt fall rapidly once the technique becomes more widely-used, as it will be, initially for commercial purposes. A second snag is that at present it is a "read-only" system: once the disc has been made, there is no chance of altering any part of it, so it has to be perfect first time. Within a very few years, however, it may be possible to record video discs using an ordinary video camera, and this will open up new possibilities for home-made interactive video.

Interactive video in Britain is still largely in an experimental stage, but in the United States it is more widely used. There is even a language video, called *Montevedisco*, for students of Spanish, which gives the user a choice of a male or female role, and takes different routes through the (adventure) programme according to the learner's choice. Perhaps of more interest to ESL learners would be an English adaptation of an idea described in *Computers and Reading Instruction* (Geoffrion, 1983). DAVID "was developed to improve the language skills of hearing-impaired college students", say the authors, and

> One DAVID lesson is a simulated employment interview used to teach communication skills. The video segments are filmed as though seen through the eyes of the person interviewed. It begins with entering the office of the personnel manager, who, after greeting the viewer, asks questions about the viewer's interests. The video-cassette shifts to a different segment based on the viewer's answer to each question. The resulting interview is remarkably true-to-life.

With all the sophistication of a good computer program combined with the realism provided by video film, interactive video must surely be a pointer to the future development of computing in education.

Programming with the chips of the future

Seymour Papert argues (Papert, 1982) that already in educational computing

> We are in the process of digging ourselves into an anachronism by preserving practices that have no rational basis beyond their historical roots in an earlier period of technological and theoretical development.

The fact that computer programming for language-learners at present largely relies on amateurs for its output must not mean that CALL fails to respond to new developments in programming techniques which will become possible as a result of improvements in hardware and "firmware" (silicon chips).

The forgiving program

The present generation of CALL programs tends to be prescriptive, in that whatever input the user types at the keyboard is either right or wrong, with no halfway-house for "nearly right" answers. Even simulations, in which there are often no right or wrong answers, nevertheless reject an input because it fails to conform to the "correct" spelling, or the "correct" syntax, as judged by the standards of the programmer. This is simply not good enough for a language-learner, who will not always be in a position to determine without assistance what kind of error is leading to the rejection of the attempt. Programs of the future must encourage extended communication with the user, allowing degrees of correctness to emerge.

This does not necessarily mean a machine that will correct every error (unless that is what the user wants!), but one that can be programmed to be "intelligent" enough to recognise when the user is making sense, and when not; a machine that can make a relatively informed guess at what the user really meant, and if there is a degree of doubt, ask "Did you really mean this?" or "Can you suggest an alternative for this word?"; a machine that can be forgiving when it comes to small errors of spelling or grammar; a machine that, in cases where precision is expected, can guide users towards a correct input, instead of merely telling them that they are wrong.

This approach to programming is alrady possible in a limited way, using present technology and programming techniques. Provided that the programmer stays within narrow language limits, a program can be fed with rules which enable it to make an assessment of an input, and, if the input does not correspond exactly with what is expected, attempt to match it with its nearest known equivalent. The important factor is the narrowness of the "microworld" which the program "learns" about. The narrower the linguistic limits, the easier it is for the programmer to anticipate likely aberrations on the part of the user, and include appropriate routines to handle them in the program. But the more extensive the "microworld", the more things can go wrong, become ambiguous, or be open to misinterpretation, and the more complex the program required to handle them.

Artificial intelligence

New developments in "natural language" systems are taking place in the business world, and are closely related to developments in "artificial intelligence". A paper on *Artificial Intelligence and Computational Linguistics: Implications for CAI,* given by Ali Farghaly (of the University of Alexander, Egypt) to the TESOL Convention in March 1984 (Farghaly, 1984), pointed out the potential benefits of these developments to Computer Aided Instruction (CAI is the American name for CAL, but with the emphasis on Instruction):

> As teachers, our judgements on students' performance concentrate on two aspects: a) students' proper use of the linguistic rules of the foreign language; b) the validity of students' statements in relation to a text, discourse or facts of life. Our objective in CAI is to produce intelligent software that can perform both tasks. Computational linguists developed efficient parsers that model human grammatical competence. Meanwhile, Artificial Intelligence scientists successfully developed systems for database retrieval and processing for the generation and analysis of texts. Incorporating such techniques into CAI will result in smarter CAI software.

The relevance of database retrieval in "natural language" systems may not be immediately obvious unless it is understood that these systems operate by searching a database composed of language-rules combined with semantic information in order to make a decision on the probable meaning of an input.

Full-scale artificial intelligence (AI) may be going a bit too far when it comes to CALL. AI not only makes guesses after consulting its database of language-rules, but is prepared to modify those rules in the light of experience; a sensitive AI system faced with a wrong spelling for the first time may be able to guess its correct version, but if the same word is misspelt in the same way on subsequent occasions, a true AI program will eventually come to the conclusion that the wrong spelling is a valid alternative, and will absorb it into its database. AI in language-learning could therefore lead to programs which "learn" all the linguistic solecisms of their users.

Even if CALL does not demand the ultimate in artificial intelligence, it could nevertheless make good use of a "natural language" system which could make rapid decisions about language input based on a comprehensive overview of the way language works. Two of the stumbling-blocks so far have been the memory-size of computers, and the slowness of the decision-making process, reliant as it is on searching, comparing and rejecting thousands of possibilities before it comes to a decision. Both of these stumbling-blocks are on the verge of being removed.

The development of new types of silicon chip, in which four or five times the capacity of a large mainframe computer can be contained in an area no bigger than a match-head, brings much closer the advent of computers the size of today's home micro with the same power as equipment which today would cost millions of pounds. This will relieve the problem of memory constraints.

The "string comparator"

Already with us is a chip dedicated to making decisions about language on a scale hitherto undreamed of — a tiny piece of silicon capable of comparing "strings" of characters at a rate of 30,000 comparisons every second. The example offered by the chip's manufacturer is a comparison between:

The quick brown fox jumps over the lazy dog.

and

Quick brown foxes tend to jump over lazy dogs.

The new chip could compare these two character-strings on an alphabetic level, a phonetic level, a semantic level or a structural level, according to the choice of the programmer. And in 75 millionths of a second, it would come up with its conclusions about the degree of similarity between the two. The use of a "string comparator" in programs with a dictionary-content, and in particular in CALL where in-depth analysis of strings can make for very slow operation, would be electrifying.

Add to the string comparator a sensitive speech-synthesiser with good intonation, and a speech-recognition device capable of handling ordinary human spoken language and turning it into written language, and the result will be a "language machine" par excellence. Though perhaps not this century.

Chapter 5

Teacher-involvement and program-development

A questionnaire covering wide-ranging aspects of CALL was sent out in January 1984 to

- ☐ 23 ESL Centres in Adult Education Institutions (AEIs) within the Inner London Education Authority (ILEA) area;
- ☐ 16 ESL Centres in Colleges of Further Education within the ILEA area;
- ☐ 39 Adult Literacy Centres within the ILEA area;
- ☐ 13 EFL Centres within the ILEA area;
- ☐ a small selection of centres outside London where CALL was known to be already in use.

The questionnaire was designed to establish:

- ☐ what interest in CALL already existed in the general area of language work in the ILEA; for this purpose, literacy was considered to be "language work";
- ☐ what makes of equipment were already in use for CALL, or might be made available to teachers if they wished to introduce their students to CALL;
- ☐ what programs were being used for CALL, and what sources had been tapped to obtain them;
- ☐ what kind of programs had succeeded and which had failed, with teachers' opinions on the reasons for their success or failure;
- ☐ what were the opinions of teachers who had already been involved in CALL as to its relevance and importance in language work;
- ☐ what were the opinions of teachers who had not yet been involved in CALL as to its relevance and importance in language work;
- ☐ whether the views of teachers in the three different educational areas — ESL, EFL and Adult Literacy — were sufficiently similar to support the theory that programs developed for one area of work could be used unchanged, or with little adaptation, in either of the others.

The small group of language centres outside London was included as a control group, mainly to discover whether there were major differences outside London in the makes of equipment in use, and whether this had affected the range of programs which could be offered to students.

EFL, ESL and Literacy — different requirements?

There were widely differing degrees of response from the ILEA centres which received the questionnaire:

	Response
ESL Centres in Adult Education Institutes (AEIs)	45%
ESL Centres in Colleges of Further Education	13%
Adult Literacy Centres	40%
EFL Centres	23%

Judging from the replies to the questionnaire — including not only the number of responses, but also their content — it seems that AEI and Literacy teachers have a far greater inclination towards the use of computers in their work than teachers based in FE Colleges. The FE responses, while not antagonistic, were more ready to question the need for introducing yet more technological wizardry into the classroom.

This variation in response may be explained in a variety of ways. The enthusiasm of AEI teachers and adult literacy groups may simply be a cry from the relatively resource-starved sectors of language-work for more imaginative tools to work with; the corollary of this is that teachers in FE, with a better range of facilities at their disposal, are more blase about adding yet more.

Another explanation may be that EFL teachers, in particular, see computers as just another way to implement drill-and-practice techniques, while the other groups are more alive to the creative opportunities which micros can open up.

It may also be true that the different styles and teaching-techniques in these three areas of language-work are not equally suited to the introduction of the computer, which is essentially a group-based rather than a whole-class activity.

The greatest enthusiasm was evident in the responses from those who had actually used computers with language-students, indicating that, given the opportunity, both teachers and students found the use of the micro stimulating.

The needs of adult literacy groups, as perceived by literacy teachers, are very different from those of second-language learners, and will therefore receive separate consideration, although some of the following comparisons between EFL and ESL work apply equally to literacy.

CALL in EFL and ESL

The differing nature of EFL and ESL classes makes it likely that they will wish to approach the use of computing in different ways, but this need not necessarily mean that the two groups will need different programs. In fact the questionnaire returns suggest that EFL and ESL could profitably learn from each other's methods, and use each other's programs almost interchangeably.

The main divergence between the two groups is in the growing provision of "computer-awareness" and "computer literacy" courses for ESL students. This may result in a situation in which EFL regards the computer primarily as a tool to assist in the learning of English, whereas ESL will be introducing computers into the classroom for their own sake, with language-learning programs tacked on as a secondary activity.

Computer literacy

Trying out new ideas in ESL work (Ives, 1984) is an account of how a computer option can be provided for language-learners, and the many benefits this may have, for teachers and students alike. The question has sometimes arisen in EFL circles as to whether the time taken in "learning the machine" can be justified, or whether it is simply a waste of good language-learning time. This article amply answers that criticism, and makes a good case for providing such short courses for ESL *and EFL* students. It will undoubtedly be more easily done in ESL, however, as there is a much greater degree of co-operation between ESL and other departments in the provision of linked skills, whereas EFL would more often than not have to rely on whatever computing expertise could be drummed up amongst language-teachers.

Frances Weinrech, a tutor on the Islington computer option, writes:

> Somehow, the combination of following instructions, both written and oral, and performing practical tasks seemed to provide a formula to break down all barriers . . . .
>
> There were problems, thank goodness, otherwise there would not have been the opportunity for students to ask for advice and discuss alternative approaches within their groups. During the first few days, I noticed how laboured the questions were, but since the subject proved so compelling to students and tutors, any shyness or diffidence was soon overcome. By the middle of the second week, confidence in spoken English had increased amazingly and we were surprised at the complexity of the language being used.

Helen Bishop, the Course Organiser, added:

> Neither of the ESL tutors had a head start over the students . . . One advantage of being a novice was that we could genuinely ask students how they were able to do something on the machine. This provided real spoken English practice as students produced detailed explanations of what went wrong and how they had tried to correct their program . . .
>
> From the language point of view, it forced the students to understand and produce complex sentences with clauses such as: "You must do this before doing that", corrective or criticising English, viz: "We should have done this" and "We forgot to do that". Working in groups of three or four,

> they were forced to listen to each other and to work out the problem together.

A course such as this can be used by students as a precursor to vocational training in computing, and also meets the needs of students who simply want to learn about computing for its own sake.

Language-learning programs

Those with no interest in computing per se (and that includes both students and teachers) can still use and benefit from programs written specifically for language-learning, and some EFL and ESL students are already using computers in this way, despite the relative dearth of suitable programs.

At least one AEI has run a computer-assisted ESL course, and the same Institute has also organised weekly courses in practical computing for AEI staff, and one-day computer-awareness courses on Saturdays.

The bulk of language-learning programs currently being used in EFL and ESL applications come from commercial sources, with a small proportion of "home-made" products, and even fewer from ILECC. One ESL centre is using programs obtained through the host school.

By and large, the commercial programs in use were given only moderate approval by questionnaire respondents, and much of the material is clearly used only because there is a distinct shortage of anything specifically intended for EFL or ESL students. The same (small) range of programs is being used by both EFL and ESL groups.

Although within the ILEA the Research Machines 380Z and 480Z micros outnumber all other makes put together, there is very little RML-based software suitable for language-work, and this clearly leaves RML-users at a disadvantage. The most popular commercial programs in use within the ILEA area are those produced by WIDA Software for the BBC Micro, and these, originally intended for an EFL environment, were widely praised by both EFL and ESL users, with comments such as "Storyboard — this is clearly the way forward", and "We need more of these". Part of the attraction of these programs lies in the authoring facility, which allows teachers to customise the data for their own students.

Other commercial programs which gained a number of specific mentions, but also provoked considerable criticism, were simulations such as SLICK, G.B. Ltd., STOCKMARKET and SURVIVOR. Most of the simulations in use are intended specifically for business applications, and understandably teachers did not find them entirely suitable for some EFL/ESL audiences, although they may be useful in ESP applications.

Nevertheless, when teachers with computer-experience were asked to rank program-*types* (as opposed to specific programs which they may have used), simulations were given high accolades, with adventure-games following close behind, indicating that users appreciate the potential of these program-types, without necessarily approving of the examples which are currently available.

Teaching and testing programs, structure practice and vocabulary-work were given a poor rating by present micro-users — although some of these activities (particularly vocabulary-work) were well-received when disguised as "games". Almost everyone with experience of computers in the classroom was more enthusiastic about word-processing than about any other single application.

There was little difference between the opinions of ESL and EFL groups, either in their judgements of what programs are available now, or in their perceptions of what kinds of program would be most useful to their students. The more important differences were between those who had used a computer in the classroom, and those who had not.

The types of work which *potential* computer-users feel is suitable for computer applications include drill-and-practice and programs to reinforce classroom lessons, whereas those with experience of the computer in the classroom favour the less prescriptive programs where inter-group communication is an important by-product. Surprisingly, most respondents did not rate the computer very highly in remedial applications.

Program features

No real conflict of interest emerged in the opinions offered about program-features.

Most teachers attached great importance to the facility of allowing the user to recap earlier parts of a program at will, where this is appropriate, although only the EFL respondents felt that a "HELP" facility was very important. (This may be a reflection of a slightly greater inclination of EFL respondents to think of the computer as a drill-and-practice-machine). There was a fairly widely-held view that "reward" sequences, such as smiling/frowning faces and music, were, at least for adults, not a useful addition to a program, and while they might be considered fun at first, students quickly tire of them.

To meet the requirements of teachers and students, programs should recognise, and preferably correct, errors on the part of the user — that is, software should have a certain amount of built-in intelligence when it comes to handling, for example, spelling mistakes. The idea of incorporating a dictionary within programs was appealing to both EFL and ESL respondents. Both groups felt that using graphics for a purpose — e.g. demonstrating concepts of length, weight, colour, etc — made more sense than graphics for their own sake.

The element of program design which found least favour was the emphasis on competition, scoring and rewards which is found in many commercial programs (although a competitive element for the purpose of increasing motivation was felt to be desirable).

The provision of facilities for storing details of a student's performance for later assessment by teachers also came in for considerable criticism, and was generally felt to be "sneaky". One respondent, however, felt that the assessment facility was useful, because in his particular situation, groups of students were obliged to use computers in one room while the teacher taught in another room, and so had no other means of knowing what use had been made of the machines, or what problems had arisen. Both groups placed considerable importance on the student's being able to take away a print-out of work done, where appropriate.

Asked about methods of error-correction, the incorporation of compulsory practice material within the program was totally rejected, and almost everyone disliked the idea of recapping "wrong" answers before exiting the program. By far the most popular methods of error-handling were the incorporation of extra practice on a *voluntary* basis, and the option, within the program, of pointing the user in the direction of further practice, which may be in the form of another program, or possibly a book, a work-card, or a teacher.

Reading, writing, listening and speaking

Of the four basic skills of reading, writing, listening and speaking in relation to computer-aided language-learning, the obvious winners in most people's estimation were reading and writing. Hardly anybody felt that "listening skills" were specially suited to the computer, but this probably reflects the very tiny number of respondents (only two!) who had actually had the experience of combining computer and audio cassette.

Few inexperienced respondents rated the computer highly with regard to "oral skills", although those who had already been using computers with language-students mentioned this most frequently as a prime reason for using simulations, games and adventures. There was a clear division of opinion between non-users on the one hand, who appear to relate "oral skills" to the ability of the student and machine to communicate orally *with each other,* and those with classroom experience of micros, who regard the computer as an important catalyst of oral communication within a group of users. Again, the dichotomy was not between ESL and EFL, but between experience and inexperience.

Functional programs

In almost every respect, ESL and EFL teachers' requirements coincide, and it seems that programs created for one group might equally well be used by the other. There is, however, one area in which EFL programmers can learn from ESL experience, and that is the area of functional applications. Many early programs written specifically for the EFL market — in particular, word-games — concentrated on out-of-context language-practice. There is a greater tendency now to place program material in an appropriate context, and authoring facilities enabling teachers to use their own choice of text or vocabulary have gone some way towards providing programs which fulfill a functional need as well as a purely linguistic one. Such programs are more likely to be acceptable to both language-learning communities.

One program-type which can fulfill both functional and linguistic purposes is the simulation. Up to now, few simulations are acceptable linguistically for either EFL or ESL groups, but from the point of view of subject-matter and function *no* commercial simulation yet meets the needs of ESL students. Databases will also be more successful if they fulfill a genuine need: a group of students could, for example, set up a database for establishing social security entitlement for students in a variety of different circumstances.

Even in games, a judicious choice of subject-matter can ensure acceptability in both camps. One example will suffice. A simple program, involving a sum presented on the screen in the normal format (for example, "24 + 12 = "), required the student to type in the words representing the answer ("thirty-six"). In an EFL class, students are taught to write numbers at an early stage, and such a program provides practice of a rather mundane subject in a reasonably stimulating way.

However, the immediate response of an ESL teacher to this program was: "In what circumstances do people actually *need* to be able to write numbers?" The most obvious answer is that most people only write the word-equivalents of numbers on cheques, so why not convert the program into a cheque-writing exercise?

As a result of this suggestion, the program was re-written, presenting an accurate graphic representation of a cheque on the screen (the features of which may be used to illustrate a discussion about banking), writing a random sum in numbers in the "box" on the cheque, and asking the student to "complete the cheque" by writing in the words. As a bonus, the computer sets up a "bank account" for the student, crediting or debiting the account according to the right or wrong answers given, and at the end of the program presents the user with a "bank statement" showing debits and credits, and the balance remaining. The resulting program is more useful and more attractive to both ESL *and EFL* sectors than the bare-bones arithmetic program.

Computers in Adult Literacy

The cheque program may be equally well used by literacy students as by ESL or EFL groups, and indeed there are programs from various sources which can be used for literacy purposes, either as they stand or with minor alterations. There are, nevertheless, special needs in literacy, and little attempt has yet been made to cater for them. The spin-off of writing programs specifically for literacy users is that many of them would be welcomed with open arms by EFL and ESL users, too, and in certain circumstances by mainstream education at every level.

Of the program ideas mentioned by literacy teachers, there are none which would not be equally applicable to EFL or ESL situations. The possibilities include:

☐ strengthening concepts of alphabetical order;

- keyboard familiarity, with special attention to upper/lower case distinctions;
- practice in the reading of instructions;
- practice in filling in forms;
- familiarisation with the highway code;
- identification of road-signs, washing symbols and other examples of graphics used to replace language;
- use of timetables and maps;
- databases containing, for example, health and fitness tips;
- interactive programs dealing with social skills.

Throughout the discussion so far, the importance of the computer in promoting verbal communication has been stressed. This use of the micro is less important with literacy students, most of whom are native speakers of English, and the emphasis is firmly back with the skills more traditionally associated with computers — the skills of reading and writing. But in literacy, more even than in ESL, function becomes an overriding factor.

Although the questionnaire-returns indicate a great untapped interest in the use of micros for literacy work, most of the respondents were only in a position to say "Please tell us what we can use!" The range of materials for ESL/EFL is poor. The range for literacy is practically non-existent. The one literacy group in the survey where micros are currently being used is using home-made materials and those obtained through liaison with the college's psychology department.

Those who answered the questionnaire nevertheless had strong views about their requirements.

Where EFL and ESL teachers saw a use for a dictionary incorporated within a program for easy reference, literacy respondents naturally prefer their students to use a real dictionary.

The use of graphics for any purpose at all was less popular amongst literacy respondents than the other groups, which is only to be expected. Whereas second-language students can make use of graphics displays to illustrate concepts, colours or contexts, native speakers do not usually need this kind of assistance. More important, many of the materials for literacy are aimed at the very young, and these tend to be avoided in adult literacy circles; so graphics must be strongly adult-oriented, otherwise they may easily be seen by students in the same light as the "childish" illustrations in "early reader" books.

Adult literacy teachers could make use of material written for the very young, provided that:

- it did not patronise the adult user by incorporating child-based graphics or rewards:
- it permitted data to be changed by the teacher.

On the other hand, infant and early-primary teachers, while wanting the facility of being able to alter data at will, often prefer reading materials which offer lively graphics and sound.

One answer to this is to write a range of literacy programs which could be adapted (via an installation program run by the teacher prior to using the programs with students) to include or exclude graphics, rewards and sound, and would also allow teachers to type in suitable vocabulary or texts. Although this would seem to give the programmer a great deal of work in writing graphic routines which some users will never require, it would on balance be an economic way of providing programs which could easily be tailored to suit a specific audience, which might range from infant-school children via secondary remedial pupils to adult literacy or EFL/ESL students.

Program customisation for literacy

Another feature mentioned by a number of literacy teachers is related to the provision of an authoring facility, but adds a new dimension to it. The use of an authoring facility normally requires some advance preparation by the teacher of new data to be entered, usually followed by the running of a program (separate from the student's program) to enter the data, after which the new data appears on the menu of options when a student uses the program at any time in the future.

The special circumstances of literacy groups call for a less formalised approach, in which the "authoring system" might simply consist of a facility for entering new data within the student's program on a "one-off" basis, the "personalised" data being created for or by a particular student or group of students, and lost when the program is finished. One program which could benefit from this approach is the "jumbler" type, as literacy teachers often use jumbling as a pencil-and-paper technique, using short, off-the-cuff sentences to suit particular students. The kind of flexibility which will allow easy customisation of programs is extremely important in literacy, but could also be useful in any language-learning or remedial situation where the student/teacher ratio permits its use.

However, the student/teacher ratio is likely to be the stumbling block when trying to devise programs which suit both literacy students and language-learners. One of the objects in computer-programming for EFL/ESL is to create programs in which the students may be as far as possible independent of the teacher. There is a certain emphasis on "marking" the work done by the student, on judging performance, on scoring and competition.

Literacy students work in much smaller units, if possible on a one-to-one basis for part of the time, and any element of pressure to respond, timing of responses, or "below-the-belt" attacks on student errors is anathema to literacy teachers and students. It is important in literacy programs to provide a means of exit from the task at all stages, so that students do not feel under pressure to continue work which they feel they cannot cope with.

It is equally important for the machine not to sit in judgement on the student's performance — to allow the *teacher* to comment on performance rather than the computer — and for the program to progress according to the speed and inclination of the student.

These requirements suggest that programs for early stages of adult literacy may need to be specialist productions, while later stages may converge with EFL/ESL to the extent that some programs, especially the ones with authoring facilities, will be interchangeable.

Word processing

One surprising finding from the questionnaire was that, of all the groups questioned, only the literacy respondents generally felt that word-processing was *not* a particularly useful application, although this opinion may change if more literacy groups get the opportunity to experience the advantages of word-processing for themselves, and also if a more suitable word-processor were available. The language-processor described in Chapter 3, with the option of magnified characters, may be of some help in literacy work.

One group of literacy teachers, however, was especially interested in word-processing, as their students often have a psychological barrier which prevents them from making progress with handwriting, whereas they have no problem with handling a keyboard.

Literacy is an area of language-work in which exploration of alternative methods of putting information into the computer could pay dividends. A "concept keyboard" — a touch-sensitive pad — can be used as an alternative to the full keyboard where very simple inputs are needed. Another possibility is the "light-pen" — a special pen which can communicate with the computer when it is used to "write" directly on the screen.

A light-pen program called LETTER FORMATION, written for young children, is a model of what might be done in this field for adults. LETTER FORMATION draws lower-case letters in red on the screen, so that the user can see the strokes of the "pen" as they are made. The child then "copies" the letter by drawing over it on the screen with the light-pen. As the shape of the letter is traced, the light-pen leaves a trail of "green ink" behind it. If the tracing starts in the wrong place, or takes a wrong direction, the flow of green ink stops. The program divides the letters into four groups (emphasising the child-based nature of the program) — "little stick", "big stick", "angled" and "round" letters. A similar program for adult users might add upper case and cursive script to the basic program idea.

Student involvement

Of all those who filled in the questionnaire, only literacy teachers mentioned the desirability of involving students in program design and production. One teacher wrote:

> I have answered this questionnaire from *my own* point of view. I feel that students too would like to present theirs before programs are produced on their behalf.

In the present period of exploration, many of those in the field of education — both teachers and students — have little idea of how computers might be used in language-learning. On the other hand, at the present time students of all kinds often know more about computers and programming that their teachers do. When more people have seen pilot projects involving language-handling, and have first hand experience of the kind of things the computer can be asked to do, there will be an explosion of ideas from all sides, especially from students (and not only literacy students) who can see ways of improving what has already been produced, and ways of extending the range of program-types to meet their needs more fully. Moreover, many students will have the necessary programming expertise to put some of their ideas into practice.

Even a comparatively small body of programs would act as a catalyst for this kind of activity, as is borne out by the increasing numbers of schoolchildren who, having seen the educational programs produced for them by adults, have now set about producing their own. While there is little chance of the large commercial publishers consulting the consumer before producing software, there are excellent reasons why programs initiated by local authorities or local programming groups should not only involve the consumer at the trialling stage (as they already do to a considerable extent) but should draw students into programming teams at the planning level, too.

What teachers want

According to the sample of ILEA teachers who responded to the CALL questionnaire, few language or literacy teachers would oppose the introduction of CALL. Most respondents claimed that the majority of their colleagues were enthusiastic in varying degrees about CALL: at worst, a small proportion were indifferent or slightly hostile. In places where CALL was already in operation, respondents reported that interest among colleagues had either remained steady or improved since its introduction; no respondents reported any hostile reactions.

Analysis of the returns also strongly suggested that those with experience of CALL in the classroom had a wider appreciation of its potential value to students, and in particular its beneficial effects as a stimulant or catalyst of conversation and exchange of ideas. Non-users tended to think of the computer as a one-to-one drill-machine; users viewed it as a flexible group-activity.

In-service training

One factor which emerges time after time in conversation with teachers is the need for in-service training (INSET) in the handling of computers, and there is general agreement about what the training should accomplish.

In the sphere of general computer literacy, in-service training should

☐ *give teachers an adequate knowledge of the mechanics of handling a* computer: how to set up the machine, switch it on and get started;

☐ provide information about program storage, and the handling and use of discs and cassettes;

☐ explain some of the problems surrounding the lack of compatibility between computers (even different models of the same make!), and the consequent lack of portability of programs from one machine to another;

☐ explain what might go wrong, and how to recover from errors.

In addition to the "general education" outlined above, prospective CALL users expect in-service training to

☐ allow time for a range of CALL programs to be demonstrated, and then to be discussed and used by the trainees;

☐ show how CALL may be integrated with other classroom activities;

☐ demonstrate how non-CALL programs may be used in a CALL environment — for example, database and word-processing programs.

CALL programming — a cottage industry?

Various means have been discussed of obtaining programs from non-CALL sources for the use of language-learners, in order to extend the present very small range of software, but ultimately the bulk of programs must be purpose-written, if they are to satisfy both students and teachers.

Yet financial backing from the Government for the Microelectronics Education Programme (MEP) is being gradually withdrawn, just at the time when expansion into EFL provision was being considered.

What little software already exists in the field has been written mainly by a few self-taught teacher-programmers, and has done no more than whet the appetite for more — and more far-reaching — programs. The question must be asked: will teacher-programmers be able to fulfill all the needs of language-students in the future?

Furthermore, what software there is, is available on a variety of different machines, so the chances are that the software you like will not run on the hardware you have to use.

These three inter-related problems of finance, programming expertise and hardware compatibility must be solved if sufficient high-quality programs are to be provided to persuade educational establishments that it is worthwhile to fund micros for language-work.

CALL has got off the ground mainly because of a few enthusiasts who saw its potential in the early days of microcomputing, and it has therefore become a kind of cottage industry, with software supplied in the main by programmers who are "amateur" in the sense of being primarily teachers, and only secondarily programmers. The amount of software which can be written in this way is limited, and the problem of who should assume responsibility for CALL programming in the future has not yet been seriously considered, although it will affect to a considerable degree the *types of program which will be offered to language students, and will* consequently affect the attitudes of language-teachers to the use of the micro in the classroom.

This lack of good software applies not only to CALL. A sixth-form student, writing in *Practical Computing* (Lansdell, 1984), pinpoints the difficulties in mainstream education. Because of the paucity of programmers in education, he claims,

> . . . educational software for the machines is purchased hastily, seldom lives up to expectations and hardly earns the titles "interesting" or "creative". The school is faced with both a lack of software and adequate knowledge, and so the expensive computer equipment lies dormant, not justifying the money invested in it.

The tiny amount of money made available by the government for program-creation through the MEP compares very unfavourably with the vast amount poured into providing schools with hardware. As a result, Britain has a fairly healthy micro industry, but little serious software to run on the thousands of British micros which have found their way into schools through government subsidies. The consequence of this is a bandwagon of software companies producing "junk" programs for mainstream education, and a very small amount of satisfactory material, mostly written or at least planned by the teachers themselves.

The bandwagon has not started rolling yet in programs for language-work, which explains the very small quantity of software in the field, largely the work of teacher-enthusiasts. One of the reasons why publishers have not responded is that they, too, are part of the vicious circle: language-teachers will not buy equipment because there is too little software, there is too little software because the purpose-written programs are spread thinly over a variety of machines, and commercial publishers are unwilling to plump for a particular machine until it shows signs of becoming "standard" in the language-industry.

A CALL watchdog

The shortage of commercially-produced software is not necessarily such a bad state of affairs. It certainly allows time for the teachers involved in language-work of various kinds to find a solution to the problem before one is imposed on them by the large publishing-houses. What is needed is an authoritative body of teachers with the support of influential LEAs, who can thrash out a scheme of future developments.

One of the priorities of such a body would be to decide as objectively as possible what range of machines to concentrate on, taking into account such factors as price, future support, and what equipment is already in use, and then to encourage publishers and independent programmers to produce a version of each published program for each of the machines selected. This would make the choice of equipment much easier.

The next step would be to establish software standards for language-based programs, so that programs may be objectively assessed in terms of their suitability for the target audience. These software standards would have to cover both programs written expressly for the target audience and software intended primarily for native-speakers but suitable or adaptable for the target audience.

The prime task of the group would then be to assemble and keep up to date a software directory, the purpose of which would be to make known to teachers what programs are available in the field of language-work, with details of sources, prices, equipment needed and other relevant data; and to offer an objective assessment of their suitability. This would serve a dual purpose: it would provide a means by which teachers could readily discover what is currently on the market, and it could also be influential enough, with the right kind of backing, to ensure that programs produced commercially were of a suitably high quality.

Who should develop CALL programs?

Few teachers (of any subject) with experience of micros in the classroom would dispute that programs written by teachers are more likely to be educationally sound that programs written by programmers. On the other hand, a professional programmer is more likely to turn out a polished product, and to be able to cope with the more advanced techniques of programming. Higgins and Johns (Higgins & Johns, 1984) comment on the dissatisfaction felt by teachers with programs which "deal exclusively with word or sentence level meaning and ignore context and discourse features". The conclusion they draw from this dissatisfaction is that:

> Naturally enough, language teachers tend to put the blame on the machines. What they fail to realise is how unaware the computer specialist is likely to be about what happens inside a language classroom and how language teaching has changed in the last twenty years. This situation will be put right only when there are enough language teachers with substantial classroom experience who also understand computers and can use their professional knowledge to exploit the machine's potential.

Another interpretation might be that a judicious combination of teaching expertise with programming expertise in the form of "programming teams" would solve the problem faster and more economically.

Some language-teachers have developed an interest in programming, but they are few in number and have little chance of satisfying the needs of the growing market for language programs. An aggravating factor in this is that most of the teacher-programmers are first and foremost teachers, for whom programming is an enjoyable spare-time hobby. There are, however, many teachers whose interest in the computer is purely as a teaching aid, and who do not have, or want to have, programming expertise, but who would still like for pedagogic reasons to have a substantial say in the development of the programs their students will be asked to use.

Higgins and Johns go on to argue that programming can be a stimulating experience for a teacher:

> A further benefit can arise from the very nature of a computer program, which has to be completely explicit in instructing the obedient but stupid machine what to do. It is, therefore, an excellent test-bed for the teacher's linguistic and pedagogic insights. The job of writing a program and trying it out can lead to realisations that the assumptions we make about how language works, and how it is taught and learned, may need to be looked at afresh.

This is indisputable. It is not an argument for teaching computer-programming to all language-teachers. It is, however, a good argument for insisting that computer-programmers of language-based programs should *either* be language-teachers themselves *or* should be part of a programming-team with considerable language-teaching experience. A further advantage of a team approach is that it may also involve the consumer — the student — as well as the teacher and the programmer.

A self-taught teacher-programmer may well unwittingly be perpetuating the treadmill envisaged by Seymour Papert (Papert, 1982), in which computers are used for the purpose of "force-feeding indigestible material left over from the pre-computer epoch". This may happen partly because this is the easiest kind of programming to learn, and most amateurs do not develop beyond this stage, and partly because a teacher's duties do not leave much time for developing complex computer-projects. With a professional programmer to handle the coding, the teacher is free to be an instigator of ideas. And if the professional programmer also has language-teaching experience, that is an added bonus.

Those of us who are "computer-holics" and need our regular "fix" of programming might feel that this approach would kill much of the pleasure in program-development. The needs of the consumer, however, must be paramount, and demand a higher rate of productivity that can be provided by the present "cottage industry". The programming team is one way of meeting the need. It will not prevent lone programmers from continuing to produce their quota of programs!

The team approach has worked well in other areas of education, for example in programs produced through the MEP and the Inner London Educational Computer Centre, and there is no reason why it should not work equally well in the production of programs for language students, given adequate financial backing.

The trouble with democracy is that it takes time. Daniel Chandler, writing about the shortage of software in mainstream English (Chandler, 1983) comments:

> There are several reasons for the shortage of suitable software. Firstly,

> commercial software publishers have little idea of what English is about. Secondly, too few English teachers are involved in local program development groups . . . Thirdly, existing projects developing software for English are only too well aware of the dangers, and producing a good learning package (of which a computer program is only one item) with part-time teams can take from one to two years.

One of the dangers of delay is that publishers with an eye to the main chance may produce "convenience software" fast, in order to fill the vacuum, and potential users may thereby be turned off the whole idea of CALL.

Is CALL a worthwhile investment?

Programming for language students will necessarily involve a substantial investment of time and money, if it is to be worthwhile. There are those who would argue that the investment required is wholly disproportionate to the value of the products, and should therefore not be made. Several factors can be set against that argument.

First, experimentation with language-based programs has hardly begun, and the output to date represents a mere scratch on the surface of potential developments in the field; future possibilities can not fairly be judged on the basis of the present corpus of material.

Furthermore, almost all the production so far is the result of "willing horses" — teachers who have had sufficient faith to invest their own time and money in an attempt to demonstrate that there is a future for language-based programs; what has been achieved so far by individual enterprise is creditable, but a great deal more could be achieved with assistance from the public purse.

There are those who will never be convinced that the computer has a useful part to play in language-learning, just as there are those who prefer a blackboard to an overhead projector, and only use a language-laboratory if forced. The Teacher's Guide to *Computer Literacy* (Lloyd, Taylor & West, 1984) recognises this attitude:

> The ability, or lack of it, to use a microcomputer will, for the next two decades at least, identify our age group and/or our talent for keeping up to date. This applies corporately as well as to individuals, and the credibility of the further education sector will in no small way depend on the speed with which colleges and their staff come to terms with IT, of which computer literacy is a major part . . . The problem for colleges, however, is to ensure *now* that time is available to provide this essential skill for all their students . . . We can only reiterate that, by describing computer literacy as a *basic skill,* we mean that all have the right to acquire this type of literacy. The responsibility of all teachers to contribute to the acquisition of this skill, must be for many one more extension of their professionalism.

This is the most cogent argument of all for the provision of specialist language programs. The Manager's Guide makes clear that computer literacy is an aspect of education which *must* be made available to all students. There is no reason why language students should be an exception to this rule. Both the Manager's Guide and the Teacher's Guide stress the importance of teaching computer literacy through programs which are relevant to the needs, experience and prospects of the students. In the case of language students, this must surely mean programs which will help them in their acquisition of the English language and/or their skill in manipulating it in written or spoken form.

Conclusion

Making the most of CALL opportunities

The possibilities now opening up for the future make it all the more exciting to be involved in CALL in its infancy. As was said in one of the BBC's DATABASE programs on television:

> The next generation of advanced super-computers will make the equipment we have today look just about as fascinating as the abacus.

CALL has special needs because of its essentially language-based nature. Earlier generations of computer hardware and software have been concerned largely with the numerical functions of computing, and it is only now that serious attention is beginning to be paid to its linguistic potential — not, it must be admitted, for any purpose of improving language-learning techniques, but for purely commercial purposes. Whatever the causes, CALL must not be slow to take advantage of developments as they arise.

Education authorities have a special responsibility here, not only in the field of CALL but across the curriculum, and at all levels. The proliferation of "educational" programs marketed in the private sector is, to put it bluntly, the result of the quest for profit. There is much chaff to sort through before the wheat can be found. The private market will not even attempt to handle areas which hold out little promise of financial reward, areas like special education, education for the handicapped, adult literacy, or CALL, and there is no reason to expect that private companies will suddenly become public benefactors in this respect. Education authorities therefore have a double responsibility, to see that *good* educational software is provided for use across the curriculum so that there is no undue reliance on the private market, and to give special attention to areas of education which have no other source of software support.

The American company Hewlett-Packard, in a booklet called *Education and Training for the New Technologies* (Hewlett-Packard, 1984) puts forward the view that:

> Many schools are now well equipped with hardware but the use of software is still limited in the study of subjects other than computing. Therefore, computer literacy is not as familiar in a wide enough sector of subjects as it should be. Although Britain is a world leader in computer availability in schools there is no place for complacency, the momentum must be increased.
>
> It must also be appreciated that computers have a wider application in subjects such as geography and particularly in remedial teaching. The limited availability of software is at present frustrating these developments.

> . . . Too often schools with good computer facilities have to rely on the extra-curricular activity of teaching staff, who themselves are inhibited by their limited knowledge of computer-aided teaching.

While there is room for dispute over the claim that "many schools are now well equipped with hardware", there is no doubt of the truth of the last assertion, that software-development is often the result of extra-curricular activity by people with no special knowledge of computing. It is a sad fact of life that much of the software developed in this way, although sound in its broad educational principles, is deficient in good programming technique to such an extent that it brings computer-aided learning into disrepute. This puts a third responsibility on the shoulders of education authorities: to take steps to ensure that, as far as possible, amateur-produced software collected and distributed by educational bodies satisfies certain minimum standards; a published "Code of Practice for Educational Programmers" might be a first step in this direction.

In the meantime, as far as CALL is concerned, software-starvation is matched by hardware-starvation. The great push towards computer literacy for all is somewhat hampered by limited resources, especially in adult education and adult literacy, where there is considerable interest in CALL, without the financial means to satisfy it.

One of the questionnaire-respondents nicely balanced the pros and cons of CALL in a letter which concluded:

> . . . there seems to be some panic involved in buying and using microcomputers. This concerns me as I was closely involved with the early developments of programmed learning and our research findings suggested that learning on machines was only succesful when teachers were enthusiastic and committed. We saw damage to confidence, etc, when new techniques were imposed.
>
> Bearing all that in mind, it would be nice if the ILEA could provide us with some computers so we could have a go!

The panic is exemplified by the fact that the offer of "cheap" computers for schools came from the Department of *Industry*, anxious to promote British-based technology, not from the Department of Education. The resultant minor flood of hardware into schools left the teachers gasping: all that equipment, with few programs to run on it, and the quality of what software there was leaving much to be desired. Would it not have been better to buy more books, more cassette-recorders, more slide-projectors? And even when software started to filter through, did all the teachers feel competent to use it, when some had had a minimal amount of training and others none at all?

These are the problems to which education authorities need to address themselves. Initiatives such as the MEP need to be extended, when in fact at the present time they are being cut back. The finance-starved poor relations of education need special help with both hardware and software. Standards must be maintained, such that computer-aided learning can thrive on good software, instead of being forced to make do with poor programs which make a mockery of the capabilities of the machine.

How public money is spent is a matter for public concern. The lives of all of us will be affected by the coming technological revolution. It is our duty to see that sufficient resources go into preparing society for the great changes that this will entail, and into taking advantage of the new technology for the greater good of the community — which means applying the best available techniques to the field of education.

To leave the last word with Seymour Papert (Papert, 1982):

> What is happening how is an empirical question. What can happen is a technical question. But what will happen is a political question, depending on social choices.

Appendix A

Acronyms

AEI	Adult Education Institute
AI	Artificial Intelligence
ASCII	American Standard Code for Information Interchange
CAI	Computer Aided Instruction
CAL	Computer Aided Learning
CALL	Computer Aided Language Learning
CET	Council for Educational Technology
CILT	Centre for Information on Language Teaching & Research
EFL	English as a Foreign Language
ESL	English as a Second Language
ESP	English for Special Purposes
FEU	Further Education Unit
FHE	Further and Higher Education
ILEA	Inner London Education Authority
ILECC	Inner London Educational Computing Centre
INSET	In-Service Training
IT	Information Technology
LEA	Local Education Authority
MEP	Microelectronics Education Programme
MSC	Manpower Services Commission
RML	Research Machines Limited
TESOL	Teachers of English to Speakers of Other Languages
VDU	Visual Display Unit
WISE	An interactive video authoring system developed by WICAT Systems

Appendix B

Bibliography

BAUM, Dr. Joan
Computers in the English class
CUNY Instructional Resource Centre, USA, 1983
Research Monograph Series: Report No. 5

BURKE, Robert L.
CAI Sourcebook
Prentice-Hall, New Jersey, 1982

CHANDLER, Daniel (editor)
Exploring English with Microcomputers
Council for Educational Technology, London, 1983

CHANDLER, Daniel
Young Learners and the Microcomputer
Open University Press, Milton Keynes, 1984

CRISPIN, Anthony,
Notes accompanying a demonstration of work in progress
Modern Languages and the new technology (conference report)
CILT/SCET, London, 1982

DAINES, Derrick
"Putting the computer in its place"
Exploring English with Microcomputers
(Ed.) CHANDLER, Daniel
Council for Educational Technology, London, 1983

DAINES, Derrick
Databases in the Classroom
Castle House Publications Ltd., Tunbridge Wells, 1984

DUGGAN, Keith
"Using EXPLOR at Highbury Quadrant Primary School"
Computers in Primary Schools, Newsletter No. 3
ILECC, London, 1984

DUKE, John
Interactive Video: Implications for Education and Training
Council for Educational Technology, London, 1983

ENGLAND, Elaine
"What does it mean to be computer literate?"
Educational Computing
November 1983

FARGHALY, Ali
"Artificial Intelligence and Computational Linguistics"
Papers of the 18th Annual TESOL Convention
TESOL, Washington, 1984

FLUCK, Andrew
"The Network in Action"
Educational Computing
April 1984

FOX, Jeremy
"Computer-assisted Vocabulary Learning"
ELT Journal
Volume 38/1, January 1984

FRASER, Shirley
"Using the Computer with infants at Beatrix Potter School"
Computers in Primary Schools, Newsletter No. 3
ILECC, London, 1984

GEOFFRION, Leo D. and Olga P.
Computers and Reading Instruction
Addison-Wesley, Massachusetts, 1983

HARRISON, Chris
"Town Planning and Shopping aid language learning"
Practical Computing
December 1982

HEWLETT-PACKARD
"Education and Training for the New Technologies"
Commitment to Britain (H-P pack of promotional materials)
1984

HIGGINS, John and JOHNS, Tim
Computers in Language Learning
Collins, London, 1984

HIGGINS, John and JOHNS, Tim
Approaches to CALL for English as a Foreign Language
(unpublished)
1983

IVES, Richard
"Trying out new ideas in ESL work"
Educational Computing
March 1984

JOHNS, Tim
Exploratory CAL: an alternative use of the computer in teaching Foreign Languages
Paper given at the British Council Computer Conference, Paris, 1982
(unpublished)

JOHNS, Tim
CALL Programs by Tim Johns
1982
(unpublished)

JONES, Ken
Simulations in Language Teaching
Cambridge University Press, 1982

JONES, Leo
Eight Simulations
Cambridge University Press, 1983

KENNING, M.J. and M-M.
An Introduction to Computer Assisted Language Teaching
Oxford University Press, 1983

LANSDELL, Jon
"Education gone mad"
Practical Computing
May 1984

LAWS, Jeff
"Linton Mead Primary School — an update from a pilot school"
Computers in Primary Schools, Newsletter No. 3
ILECC, 1984

LLOYD, TAYLOR and WEST
Computer Literacy - Part 1: A Manager's Guide
Further Education Unit, London, 1984

LLOYD, TAYLOR and WEST
Computer Literacy - Part 2: A Teacher's Guide
Further Education Unit, London, 1984

PAPERT, Seymour
Mindstorms: Children, Computers and Powerful Ideas
The Harvester Press, Brighton, 1982

POWER, Paul
"The Micro Way"
Times Educational Supplement
March 30, 1984

STEWART, Jan
"Does the use of the micro inhibit the development of language in children?"
Exploring English with Microcomputers
(Ed.) CHANDLER, Daniel
Council for Educational Technology, London 1983

WHITE, George M.
"Speech Recognition: an idea whose time is coming"
BYTE
January, 1984

WISELY, Carol
"Go West, Young Man, Go West"
Computers in Primary Schools, Newsletter No. 3
ILECC, 1984

Appendix C

Directory of CALL Programs

This directory contains a selection of programs which may be of use in CALL. The programs are offered as samples of available software which may repay investigation, and are not guaranteed as to quality or suitability for CALL purposes.

In addition to the program name, the directory contains the following information:

- ☐ publisher's code (a directory of publishers is to be found in Appendix D);
- ☐ intended target audience or age-range — though many programs may be adaptable to other teaching environments;
- ☐ availability on RML, BBC, Spectrum or other microcomputers, as far as this is known;
- ☐ a page reference, if the program has been mentioned by name in the text;
- ☐ a brief description of the program's aim or content.

The programs are arranged in sections corresponding to the categories dealt with in Chapter 1, plus two extra categories:

1. Alphabet-based
2. Letter-word manipulation
3. Word-based
4. Text-based
5. Database and rule programs
6. Structure-based
7. Adventure games
8. Simulation and role-play
9. Aids to creative writing
10. (No entries)
11. Authoring languages
12. Miscellaneous subjects
13. Content-free databases

Section 1: Alphabet-based programs

Program: Abshort
Publisher: Henry
Target: primary +
Micros: BBC, RML, Spectrum
Page: 19

Alphabet recognition. From DYSLEXIA suite of programs.

Program: Alphabet
Publisher: EdSoft
Target: primary
Micro: BBC

Alphabet sequencing

Program: Aztec and Masterword
Publisher: ESM
Target: flexible
Micro: BBC

Dictionary game, and a version of "Mastermind" using the alphabet

Program: Dictionary **Micro:** BBC
Publisher: Daco
Target: literacy

Four programs to develop dictionary skills

Program: Dictionary **Micro:** RML
Publisher: MJL **Page:** 19
Target: EFL/ESL

Game to strengthen dictionary-ordering concepts

Program: Educational Package No. 1 **Micro:** BBC
Publisher: Molimerx
Target: 9-14

Alphabetical order, use of dictionary and encyclopaedia

Program: Letter Formation **Micro:** BBC
(Startwrite) **Page:** 137
Publisher: ESC
Target: infant

Formation of the alphabet — needs ROBIN or DATAPEN lightpen

Program: Nowrite **Micro:** RML
Publisher: ILECC **Page:** 19
Target: primary

Letter-formation

Program: Order! Order! **Micro:** RML
Publisher: MJL
Target: EFL/ESL

Game to strengthen dictionary-ordering concepts

Section 2: Letter-word manipulation programs

Program: Acros **Micros:** BBC, RML
Publisher: Chelsea
Target: flexible

Acrostic, with authoring facility

Program: Add **Micro:** RML
Publisher: MJL **Page:** 134
Target: EFL/ESL

Spelling of numbers 1-100

Program: An Eye for Spelling **Micro:** BBC
Publisher: ESM
Target: 5-11

Spelling strategies

Program: Anagram Micro: RML
Publisher: EdSoft
Target: 9-14
Anagrams of 56 common nouns, with clues

Program: Bank Micro: RML
Publisher: MJL Page 134
Target: EFL/ESL
Spelling of numbers 100-1000

Program: Hangman Micros: BBC, RML, Spectrum
Publisher: Henry Page: 20
Target: primary +
Hangman game for spelling support

Program: Hangman Micro: RML
Publisher: SPA
Target: flexible
Traditional hangman game

Program: Help yourself with spelling Micro: RML
Publisher: Sussex
Target: flexible
Rules of spelling, plus strategies for memorising them

Program: Letra Micro: BBC
Publisher: MUSE
Target: primary
Game to assist reading, comprehension and spelling

Program: Magic E Micro: BBC
Publisher: Sherston
Target: primary
Use of final -e — diagnostic, teaching and practice materials

Program: Readwell Micro: BBC
Publisher: Daco
Target: literacy
Eight programs, inc. diagnostics, recognition, comprehension

Program: Snaffle Micro: Spectrum
Publisher: Longman
Target: flexible
Word-building game

Program: Spellbound Micros: BBC, RML
Publisher: GSN
Target: 8-14
Spelling game

Program: Spellwell Micro: BBC
Publisher: S/C
Target: flexible
Spelling game with help and authoring facilities

Program: Spellers Micros: BBC, RML
Publisher: Sloane Page: 21
Target: primary
Spelling game with rudimentary authoring facility

Program: Wilt Micros: BBC, RML
Publisher: Longman
Target: flexible
Hangman-type game, with authoring facility

Program: Word Skill Micro: BBC
Publisher: Chalksoft Page: 21
Target: flexible
Hangman-type game, words and phrases, with authoring facility

Program: Word Hunt Micro: BBC
Publisher: ESM
Target: flexible
Create as many words as possible from a "master word"

Program: Word Wizard Micro: Spectrum
Publisher: Longman
Target: flexible
Anagram game, with built-in dictionary

Program: Wordflash Micro: RML
Publisher: EdSoft
Target: 9-14
Spelling aids using abstract words

Program: Wordhang Micro: BBC
Publisher: Bourne
Target: 5-13
Hangman-type game

Program: Words Micro: RML
Publisher: MJL Page: 21
Target: EFL/ESL
Word-building game, with spelling help

Program: Wordspin Micro: Spectrum
Publisher: (listing) Page: 21
Target: EFL
Spelling trainer — program listing in *Computers in Language Learning* (Higgins and Johns)

Program: Wordspin — **Micro:** RML
Publisher: MJL — **Page:** 21
Target: EFL/ESL

Spelling trainer (based on Higgins and Johns) with authoring

Section 3: Word-based programs

Program: Call My Bluff — **Micro:** RML
Publisher: Macmillan
Target: 11-18

Similar to TV game, with authoring facility

Program: Choicemaster — **Micro:** BBC
Publisher: Wida — **Page:** 23
Target: EFL

Multiple-choice authoring system

Program: Crossword — **Micro:** RML
Publisher: GSN
Target: flexible

Crosswords, with authoring facility

Program: Crossword Puzzler/ 20 Crossword Puzzles — **Micros:** BBC, Spectrum
Publisher: MicroTrust
Target: flexible

Crossword authoring system and demonstration puzzles

Program: Crossword Twister — **Micro:** BBC
Publisher: Stack
Target: flexible

Scrabble/crossword combination — needs lightpen

Program: Crossword Master — **Micro:** BBC
Publisher: Wida — **Page:** 27
Target: EFL

Crossword program, with authoring facility

Program: Crossword Challenge — **Micro:** BBC
Publisher: Wida — **Page:** 27
Target: EFL

30 Intermediate crossword puzzles

Program: Gapkit — **Micro:** BBC
Publisher: Camsoft
Target: flexible

Gap-filling authoring system

Program: Highflier Micro: BBC
Publisher: GTM Page: 22
Target: primary
Homophones — 23 sets, 6 tests

Program: Identikit Micros: BBC, RML, Spectrum
(from the Dudley Programs)
Publisher: Heinemann
Target: primary
Creating identikit pictures, with vocabulary

Program: Learn to Read Micro: Spectrum
Publisher: Macmillan
Target: primary
Selection of word-game programs on four cassettes

Program: Mask Micro: RML
Publisher: MJL Page: 27
Target: teacher
Vocabulary puzzle authoring system

Program: Multi-choice Micro: RML
Publisher: SPA
Target: flexible
Multiple-choice authoring system

Program: Phrasals Micro: RML
Publisher: MJL Page: 26
Target: EFL/ESL
Multiple-choice on phrasal verbs, with authoring facility

Program: Phrase Micro: MJL
Publisher: MJL Page: 26
Target: EFL/ESL
Multiple-choice on idioms, with authoring facility

Program: Podd Micro: BBC
Publisher: ASK
Target: primary
Vocabulary extension — 120 verbs

Program: Printer's Devil Micro: Spectrum
Publisher: Collins
Target: EFL
Spelling deduction game, based on Eleusis principle

Program: Reading and Language Skills
Micro: BBC
Publisher: Macmillan
Target: 5-8
Six reading skills programs, some with authoring facility

Program: Scrabble
Micros: BBC, Spectrum
Publisher: Psion
Page: 27
Target: flexible
Computerised Scrabble

Program: Shuffle
Micro: RML
Publisher: MJL
Page: 27
Target: teacher
Vocabulary puzzle authoring system

Program: Super Square
Micro: RML
Publisher: GSN
Target: flexible
Wordsquare program, with authoring facility

Program: TES/T
Micro: Tandberg
Publisher: Hutton
Page: 24
Target: flexible
Context-free CAL authoring program

Program: Teacher's Toolkit
Micro: BBC
Publisher: Wida
Page: 24
Target: flexible
Context-free CAL authoring program

Program: The Method
Micro: BBC
Publisher: BeeBinEd
Target: flexible
Audio-linked question/answer or dictation

Program: Where/we're/wear — There/they're/their
Micro: BBC
Publisher: Primary
Target: 8-11+
Demonstration and gap-filling

Program: Word Sequencing
Micro: BBC
Publisher: Acornsoft
Target: flexible
Word jumbler

Program: Word Pairs Micro: BBC
Publisher: ESM
Target: flexible

Pelmanism — similes, antonyms, synonyms, plus authoring

Program: Word Store and Synonyms Micro: BBC
Publisher: MUSE
Target: flexible

Build your own dictionary

Program: Wordbuild Micro: BBC
Publisher: GTM
Target: flexible

Phonic word-building game, with authoring facility

Program: Words Words Words Micro: BBC
Publisher: Acornsoft
Target: primary

Illustrated, animated vocabulary building

Program: Wordsquare Micro: RML
Publisher: SPA Page: 27
Target: teacher

Wordsquare authoring system

Section 4: **Text-based programs**

Program: Back Home Micro: BBC
Publisher: Wida
Target: EFL

Multi-skills programs, based on songs

Program: Branching Story Micros: BBC, RML
Publisher: Chelsea Page: 29
Target: flexible

Branching story authoring program

Program: Cloze Micro: BBC
Publisher: LTS
Target: flexible

Cloze with authoring facility — word- or letter-deletion

Program: Clozemaster Micro: BBC
Publisher: Wida Page: 31
Target: EFL

Cloze authoring system

Program: Copywrite **Micro:** BBC
Publisher: ESM
Target: flexible
Text-building, with authoring facility

Program: Developing Tray **Micros:** BBC, RML
Publisher: ILECC **Page:** 32
Target: flexible
Text-building, with authoring facility

Program: Early Punctuation **Micro:** Spectrum
Publisher: Sinclair
Target: primary
Four punctuation marks (? ! : ,), with authoring facility

Program: Elread **Micro:** RML
Publisher: MJL **Page:** 29
Target: EFL/ESL
Spelling/observation program, with authoring facility

Program: Lingo **Micro:** BBC
Publisher: ComMs
Target: flexible
Text construction game

Program: Micro Verse **Micro:** BBC
Publisher: Wida
Target: EFL
Listening/reading skills, based on poems

Program: Micro Stories **Micro:** BBC
Publisher: Wida
Target: EFL
Cloze, text-building, dictation — based on short stories

Program: Multitest **Micro:** BBC
Publisher: A/W
Target: teacher
Multiple-choice and cloze authoring system

Program: Punctuation Pete **Micro:** Spectrum
Publisher: Hill
Target: primary
Punctuation practice at four levels

Program: Quartext **Micro:** BBC
Publisher: Longman
Target: EFL
Text-building, cloze and other text games — authoring extra

Program: Readability Micro: BBC
Publisher: A/W
Target: teacher

Reading-age assessment

Program: Readability Micro: RML
Publisher: GSN
Target: teacher

Reading-age assessment

Program: Readamatics Micro: RML
Publisher: Longman Page: 31
Target: flexible

Cloze, with authoring facility, with reading-age assessment (also available as *Wordamatics* from ILECC)

Program: Readlevel Micro: BBC
Publisher: Daco
Target: teacher

Reading-age assessment

Program: Reveal 12345 Micro: BBC
Publisher: Molimerx
Target: flexible

Text-building, with authoring facility

Program: Scanning Micros: BBC, RML, Spectrum
(from the Dudley Programs)
Publisher: Heinemann
Target: primary

40 comprehension/note-taking passages on the theme of "weather"

Program: Sentence Sequencing Micro: BBC
Publisher: Acornsoft
Target: flexible

Sentence jumbler

Program: Sentence Maker Micro: BBC
Publisher: ESC
Target: infant

Sentence building — needs ROBIN lightpen and Voice-Box

Program: Sentence Builder Micro: BBC
Publisher: ESM
Target: primary

Text creation program — needs concept keyboard

Program: Sequence Micro: BBC
Publisher: Solent
Target: flexible

Sentence sequencing, with authoring facility

Program: Speed Micros: BBC, RML, Spectrum
Publisher: Henry Page: 29
Target: primary+

Reading-speed pacemaker, with authoring facility

Program: Storywriter Micro: BBC
Publisher: ESM Page: 29
Target: primary

Branching story program, with authoring facility

Program: Storyboard Micro: BBC
Publisher: Wida Page: 32
Target: EFL

Text-building program, with authoring facility

Program: Storyboard Plus Micro: BBC
Publisher: Wida
Target: EFL

Text-building program, plus 18 original stories

Program: Textbag Micro: Spectrum
Publisher: (listing) Page: 32
Target: EFL

Text-building program, listed in *Computers in Language Learning* (Higgins and Johns)

Program: Text-builder Micro: RML
Publisher: MJL Page: 32
Target: EFL/ESL

Text-building program, with authoring facility

Program: The Storyline Micro: BBC
Publisher: Daco
Target: primary+

Assisted composition

Program: Unitext Micro: BBC
Publisher: Coventry
Target: flexible

Text-building program, with authoring facility

Program: What's your Reason? Micro: BBC
Publisher: A/W
Target: secondary

Co-operative team-game of reading and reasoning

Section 5: **Database and Rule Programs**

Program: Animal, Vegetable, Mineral   Micros: BBC, RML, Spectrum
Publisher: A/W   Page: 35
Target: flexible

Animal-type program

Program: Animal Vegetable Mineral   Micro: BBC
Publisher: Bourne
Target: 7-13

Animal-type program

Program: Edfax   Micro: BBC
Publisher: Tecmedia
Target: flexible

Electronic magazine, with authoring facility

Program: Electronic News   Micro: BBC
Publisher: NES
Target: flexible

Electronic magazine, with authoring facility

Program: Hobbies   Micro: BBC
Publisher: MJL   Page: 35
Target: flexible

Hobbies database, with authoring facility

Program: Jackass   Micro: Sharp
Publisher: Sharp   Page: 36
Target: EFL

Animal-type program

Program: Loan   Micro: Spectrum
Publisher: (listing)   Page: 37
Target: EFL

Language-generator — program listing in *Computers in Language Learning* (Higgins and Johns)

Program: Loan   Micro: RML
Publisher: MJL   Page: 37
Target: EFL/ESL

Language-generator — based on Higgins and Johns

Program: Personal   Micro: RML
Publisher: MJL   Page: 35
Target: EFL/ESL

Vocabulary of "good qualities" — game with thesaurus

Program: Prestel training pack Micros: BBC, RML
Publisher: CET
Target: flexible
Software/training materials for Prestel/electronic magazines

Program: Tele-book Micro: BBC
Publisher: 4MATION
Target: primary
Electronic magazine, with authoring facility

Program: Telesoftware Micro: RML
Publisher: RML
Target: flexible
Electronic magazine, with authoring facility

Section 6: Structure-based Programs

Program: Forsince Micro: RML
Publisher: MJL Page: 39
Target: EFL
For/since distinction — tutor-program plus practice games

Program: Lifespan Micro: RML
Publisher: MJL Page: 39
Target: EFL/ESL
Lifespan questionnaire, including the language of suggestions

Program: Photofit Micro: Spectrum
Publisher: (listing) Page: 39
Target: EFL
Comparisons — program listing in *Computers in Language Learning* (Higgins and Johns)

Section 7: Adventure games

Program: Adventure Generator Micros: BBC, RML
Publisher: Chelsea Page: 65
Target: flexible
Adventure authoring program

Program: Adventure Island Micros: BBC, Spectrum
Publisher: Ginn
Target: primary+
Adventure game

Program: Adventure **Micro:** RML
Publisher: SPA **Page:** 42
Target: flexible

Adventure game

Program: Colossal Adventure **Micros:** BBC, Spectrum
Publisher: Level 9 **Page:** 42
Target: flexible

Adventure game

Program: Dungeon Builder **Micro:** Spectrum
Publisher: Dream
Target: flexible

Adventure game generator

Program: Erik the Viking **Micro:** BBC
Publisher: Mosaic
Target: flexible

Adventure game

Program: Granny's Garden **Micros:** BBC, Spectrum
Publisher: 4MATION **Page:** 42
Target: primary

Aventure game

Program: Spooky Manor **Micro:** BBC
Publisher: Acornsoft
Target: primary

Unusual adventure game, requiring co-operation between players

Program: The Hobbit **Micros:** BBC, Spectrum
Publisher: Melbourne **Page:** 42
Target: flexible

Adventure game based on Tolkien story

Program: The Knight's Adventure **Micros:** BBC, RML
Publisher: Sloane **Page:** 42
Target: primary

Problem-solving adventure-type game

Program: The Quill **Micro:** Spectrum
Publisher: Gilsoft **Page:** 64
Target: flexible

Adventure-game generator

Program: Tombs of Arkenstone **Micros:** BBC, RML, Spectrum
Publisher: A/W
Target: flexible

Adventure game, plus adventure generator

Program: Your Adventure Micros: BBC, Spectrum
Publisher: LTS
Target: primary+
Adventure games and generator, combined with language skills

Section 8: Simulation and role-play

Program: 1984 Micro: BBC
Publisher: Incentive
Target: flexible
Simulation — running the UK economy

Program: Archaeology Micro: BBC
Publisher: CSH
Target: primary+
Open-ended simulation for authoring by students

Program: Car Journey Micros: BBC, RML, Spectrum
(from the Dudley Programs) Page: 46
Publisher: Heinemann
Target: primary
Route-planning simulation

Program: Chatterbox Micro: BBC
Publisher: BC Page: 52
Target: EFL
Simulated dialogue authoring program

Program: Farm Micros: BBC, RML
Publisher: Sloane Page: 44
Target: primary
Simple simulation, with text and bar-charts

Program: Great Britain Ltd. Micro: BBC
Publisher: Hessel Page: 44
Target: flexible
"Running the country" simulation

Program: Inheritance Micro: BBC
Publisher: Hessel Page: 44
Target: flexible
"Running a business" simulation

Program: Mary Rose Micros: BBC, RML
Publisher: Ginn
Target: 9-16
Simulation of the search for the "Mary Rose"

Program: Osprey!
Publisher: Bourne
Target: primary
Micro: BBC
Simple conservation simulation

Program: Rice Farming
Publisher: Longman
Target: flexible
Micros: BBC, RML
Simulation of rice-farming in India

Program: Slick!
Publisher: BP
Target: flexible
Micros: BBC, RML
Page: 44
Conservation simulation

Program: Stockmarket
Publisher: ASP
Target: flexible
Micro: BBC
Page: 44
Stockmarket simulation

Program: Stokmark
Publisher: Acornsoft
Target: flexible
Micro: BBC
Page: 44
Stock Exchange simulation

Program: Suburban Fox
Publisher: Ginn
Target: 9-14
Micro: RML
Page: 44
Simulation involving keeping a fox alive in an urban environment

Program: Survivor
Publisher: MP
Target: flexible
Micro: BBC
Page: 44
"Survive a shipwreck" simulation

Program: Talkback
Publisher: Acornsoft
Target: 10+
Micro: BBC
Simulated dialogue authoring program

Program: Telemark
Publisher: Acornsoft
Target: flexible
Micro: BBC
Page: 44
"Running a business" simulation

Program: The Black Death
Publisher: Longman
Target: flexible
Micros: BBC, RML
1349 Plague simulation

Program: The Co-op Game **Micro:** BBC
Publisher: Co-op
Target: flexible

"Running a Co-op" role-play — free to schools

Program: The Energy Crisis **Micro:** BBC
Publisher: SSS
Target: flexible

Simulation — keeping Britain self-sufficient in energy

Program: Tourism **Micro:** BBC
Publisher: Nelcal
Target: secondary

Simulation of developing the tourist trade on a tropical island

---

Program: Townplanner **Micros:** BBC, RML, Spectrum
(from the Dudley Programs) **Page:** 46
Publisher: Heinemann
Target: primary

Townplanning simulation

Program: Townplanning **Micro:**
Publisher: (listing) **Page:** 45
Target: EFL

Townplanning simulation — program listing in *Practical Computing*, September 1982

Program: Watching the Street **Micro:** BBC
Publisher: Gilsoft
Target: primary

Unusual simulation/database — post-computer lanugage work

Section 9: Aids to creative writing

Program: ABC **Micro:** BBC
Publisher: Acornsoft
Target: 7+

Simple word-processor with large-character viewing facility

Program: Bank Street Writer **Micro:** Atari
Publisher: Broderbund **Page:** 69
Target: flexible

Word-processor

Program: Beeline **Micro:** BBC
Publisher: ESM
Target: primary

Elementary word-processor

Program: Easywrite Micro: BBC
Publisher: System
Target: primary

Simple text-construction, using on-screen choice of words

Program: Edword Micro: BBC
Publisher: Clwyd Page: 69
Target: flexible

Word-processor

Program: Language development pack Micro: BBC
Publisher: LTS
Target: 9-15

Three programs encouraging flexible composition skills

Program: Mini Office Micros: BBC, Spectrum
Publisher: Database
Target: flexible

Cheap and simple word-processor, database and spreadsheet

Program: Simplified word-processor Micro: BBC
Publisher: Heinemann
Target: flexible

Word-processor for beginners

Program: Story maker Micro: Spectrum
Publisher: A/W
Target: 8+

Text-handling, plus limited creative writing skills

Program: Story Micro: BBC
Publisher: Daco
Target: literacy

Four programs to assist composition skills

Program: TXED Micro: RML
Publisher: RML Page: 69
Target: flexible

Word-processor

Program: Tasword2 Micro: Spectrum
Publisher: Tasman
Target: flexible

Word-processor

Program: Terrible Tales — Monsters and Giants
Micro: BBC
Publisher: Longman
Target: 8+
Text-handling and limited creative writing skills

Program: View
Micro: BBC
Publisher: Acornsoft
Target: flexible
Word-processor

Program: Word
Micro: RML
Publisher: AUCBE
Target: 9+
Assists in the learning of word-processing techniques

Program: Wordgram
Micro: BBC
Publisher: Daco
Target: primary
Creative writing via parts of speech

Program: Wordstar
Micro: RML
Publisher: Micro-Pro
Page: 70
Target: flexible
Word-processor

Program: Write On
Micros: BBC, RML, Spectrum
Publisher: A/W
Target: 5+
Text-construction and introduction to word-processing

Program: Write
Micro: RML
Publisher: MEP
Page: 69
Target: flexible
Easy-to-use word-processor

Section 10: Total-package software

No entries in this section

Section 11: Authoring languages

Program: Author
Micro: BBC
Publisher: ESM
Target: teacher
Flexible authoring system for CAL and teletext programs

Program: Easywriter Micro: RML
Publisher: NCC
Target: teacher
Simple authoring language for text-based CAL

Program: Microtext Micro: BBC
Publisher: Acornsoft
Target: teacher
CAL authoring language

Program: Microtext Micro: RML
Publisher: Transdata
Target: teacher
CAL authoring language

Program: Micro-Prolog Micros: RML, Spectrum
Publisher: LPA
Target: teacher
"Artificial-intelligence" programming language

Program: Question-Master Micro: BBC
Publisher: Hutch
Target: teacher
Flexible question-answer authoring language

Section 12: Miscellaneous subjects

Program: Accidents Datafile Micros: BBC, RML
Publisher: RoSPA
Target: school
Database for safety education

Program: All fingers go! Micro: BBC
Publisher: MicroTrust
Target: flexible
Touch-typing tutor, developed by National Extension College

Program: Balance your diet Micro: RML
Publisher: CMS
Target: flexible
Diet analysis, inc. comparison with 3rd World and ideal diets

Program: Brainstorm Micros: CP/M machines
Publisher: Caxton
Target: flexible
Ideas-processor

Program: British Towns   **Micro:** BBC
Publisher: Aztec
Target: flexible

Map with anagrams of towns, facility for adding extra data

Program: Coucapcur   **Micro:** RML
Publisher: EdSoft
Target: 9-14

Quiz on countries, capitals and currencies

Program: Entrepeneur   **Micros:** BBC, Spectrum
Publisher: Triptych
Target: adult

Serious tutorial on starting a small business

Program: Eye 1 and Eye 2   **Micros:** BBC, RML
Publisher: Longman
Target: secondary

How the eye works

Program: First Aid   **Micros:** BBC, RML, Spectrum
Publisher: NCS
Target: flexible

Illustrated first-aid study course, with test material

Program: Highway Code
Publisher: CRL
Target: flexible

Illustrated multiple-choice highway code test

Program: Hotel   **Micro:** BBC
Publisher: Pitman
Target: flexible

Simulation of the use of computers in running a small hotel

Program: Information Technology Project   **Micros:** BBC, RML
Publisher: Addison
Target: 11-14

Pack of materials suitable for introduction to IT

Program: Leaves/Pond Animals   **Micros:** BBC, RML
Publisher: Heinemann
Target: primary+

Text-based nature study identification programs

Program: Let's Explore London Micro: BBC
Publisher: CSH
Target: primary

Travelling round London by underground

Program: Living on a low budget Micro: BBC
Publisher: EdTech
Target: secondary

Life Skills material adapted from School Leavers' booklets

Program: M*N*P*L*Y Micro: BBC
Publisher: SSS
Target: flexible

Computerised Monopoly

Program: Making Ends Meet Micros: BBC, RML
Publisher: CMS
Target: flexible

Social education — financial problems of adult life

Program: Mansfield Park Micros: BBC, Spectrum
(also other titles in series)
Publisher: Sussex
Target: 15-18

Background, content, characters and themes of the novel

Program: Mastermind Micro: BBC
Publisher: Mirror
Target: flexible

Like the BBC quiz, with authoring facility

Program: Microcomputer Careers Pack Micros: BBC, RML
Publisher: A/W
Target: flexible

Careers analysis and information

Program: Micro English Micro: BBC
Publisher: LCL
Target: secondary

O-level: summary, note-taking, etc. with authoring facility

Program: Microdiet Micro: RML
Publisher: Longman
Target: flexible

Food/nutrient database

Program: Nu-Pack Micro: RML
Publisher: ILECC
Target: 11-14
Diet and nutrition program

Program: Organise Micro: RML
Publisher: MJL
Target: flexible
Ideas-processor

Program: Other Worlds Micro: BBC
Publisher: Longman
Target: primary
The world we live in — question/answer program

Program: Shirley Conran's Magic Garden Micro: BBC
Publisher: Acornsoft
Target: flexible
Gardening database, with authoring facility

Program: Summary and comprehension guidelines Micro: RML
Publisher: Sussex
Target: flexible
Summarising, note-taking, multiple-choice techniques

Program: The Computer Cookbook Micro: Spectrum
Publisher: Bug-Byte
Target: flexible
Menu-planner and recipes, with shopping list printout

Program: The Election Program Micro: BBC
Publisher: SSS
Target: flexible
Database of British election statistics, "swingometer" etc.

Program: The Music System Micro: BBC
Publisher: Island
Target: flexible
Music editor, keyboard and synthesiser

Program: Time Trucker Micro: BBC
Publisher: ASK
Target: 8-14
12-hour/24-hour clock distinction — game

Program: Touch‘n’go Micro: RML
Publisher: Caxton
Target: flexible
Touch-typing tutor

Program: Watch your weight Micro: BBC
Publisher: Acornsoft
Target: adult
Diet, calorie control and recipes

Program: What it's worth Micro: BBC
Publisher: A/W
Target: flexible
An “auction” — designed for multi-cultural classes

Program: Which salt? Micros: BBC, RML, Spectrum
Publisher: Micro
Target: secondary
Recognition of chemicals from test reactions

Program: World-wise Micro: BBC
Publisher: Bourne
Target: 7-15
Geography database

Section 13: Content-free databases

Program: Database Micro: BBC
Publisher: Acornsoft
Target: flexible
Content-free database

Program: Dataprobe Micro: RML
Publisher: Addison
Target: flexible
Content-free database — supplied with data on four topics

Program: Factfile Micros: BBC, RML, Spectrum
Publisher: CMS
Target: flexible
Content-free database — supplied with specimen data

Program: **Masterfile** Micro: BBC
Publisher: **Beebug**
Target: **flexible**

Content-free database

Program: **Microscan** Micro: RML
Publisher: **ILECC**
Target: **flexible**

Content-free database

Appendix D

Directory of Software Publishers

It is often difficult to obtain the programs you want from retail outlets. Retailers stock home-computer games and utilities, but are wary of "educational" programs, which do not sell in large quantities.

It is therefore easier to buy direct from the publisher, or ask the publisher for the address of a local stockist (if the software is supplied to retailers) where you may be able to see the program in action before you buy.

The following directory covers only those publishers whose software is listed in Appendix C.

A/W
E.J. Arnold & Son Ltd.,
(Arnold-Wheaton Software),
Parkside Land,
Dewsbury Road,
Leeds, LS11 5TD.

ASK
A.S.K. Ltd.,
London House,
68 Upper Richmond Road,
London, SW15 2RP.

ASP
ASP Software,
Number One,
Golden Square,
London, W1R 3AB.

AUCBE
AUCBE,
Endymion Road,
Hatfield,
Herts, AL10 8AU.

Acornsoft
Acornsoft Ltd.,
4a Market Hill,
Cambridge, CB2 3NJ.

Addison
Addison-Wesley,
53 Bedford Square,
London, WC1B 3DZ.

BC
The British Council,
20 Carlton House Terrace,
London, S.W.1.

BP
BP Educational Service,
PO Box 5,
Wetherby,
Yorks., LS23 7EH.

BeeBinEd
The Coach House,
Hunsdonbury,
Ware,
Herts., SG12 8PW.

Beebug
Beebugsoft,
PO Box 109,
High Wycombe,
Bucks., HP11 2TD.

Bourne
Bourne Educational Software,
Headbourne Worthy,
Winchester,
Hants., SO23 7SQ.

Broderbund
Broderbund Software Inc.,
17 Paul Drive,
San Rafael,
CA 94903, U.S.A.

Bug-Byte
Mulberry House,
Canning Place,
Liverpool, L1 8JB.

CET
Council for Educational Technology,
3 Devonshire Street,
London, W1N 2BA.

CMS
Cambridge Micro Software,
Cambridge University Press,
Shaftesbury Road,
Cambridge, CB2 2RU.

CSH
Cambridgeshire Software House Ltd.,
The Town Hall, Box 6,
St. Ives,
Huntingdon, PE17 4AL.

Camsoft
Camsoft,
7 Roman Drive,
Leeds,
West Yorks., LS8 2DR.

Caxton
Caxton Software Ltd.,
10/14 Bedford Street,
Covent Garden,
London WC2E 9HE.

Chalksoft
Chalksoft Ltd.,
37 Willowslea Road,
Worcester, WR3 7QP.

Chelsea
Chelsea College,
Hudson Buildings,
552 Kings Road,
London, SW10 0UA.

Clwyd
Clwyd Technics,
Unit 4,
Antelope Industrial Estate,
Rhydymwyn,
Clwyd, CH7 5JH.

Co-op
Co-operative Retail Services Ltd.,
54 Maryland Street,
London, E15.

Collins
Collins Educational,
8 Grafton Street,
London, W1X 3LA.

ComCon
Computer Concepts,
16 Wayside,
Chipperfield,
Herts., WD4 9JJ.

ComMs
Complete Microcomputer Services Ltd.,
32-38 Osnaburgh Street,
London, NW1 3ND.

Coventry
Coventry (Lanchester) Polytechnic,
Dept. of Language Studies,
Priory Street,
Coventry, CV1 5FB.

Daco
59 Mackenzie Road,
Moseley,
Birmingham, B11 4EP.

Database
Database Publications,
68 Chester Road,
Hazel Grove,
Stockport, SK7 5NY.

Dream
Dream Software,
18 Cromwell Road,
Basingstoke,
Hants., RG21 2NR.

ESC
Educational Software Co., (Southport),
108 Parthenon Drive,
Liverpool, L11 7AQ.

ESM
ESM,
Duke Street,
Wisbech,
Cambs., PE13 2AE.

EdSoft
Educational Software,
76 Woodville Road,
Exmouth,
Devon, EX8 1SW.

EdTech
Ed Tech Services,
The Barn, Thorney,
Peterborough, PE6 0SA.

4MATION
4MATION,
Linden Lea,
Rock Park,
Barnstaple,
Devon EX32 9AQ.

GSN
GSN Educational Software,
Manchester University Press,
Oxford Road,
Manchester, M13 9PL.

GTM
GTM Software,
864 York Road,
Leeds, LS14 6DX.

Gilsoft
Gilsoft,
30 Hawthorn Road,
Barry,
South Glamorgan, CF6, 8LE.

Ginn
Ginn Microcomputer Software,
Prebendal House,
Parson's Fee,
Aylesbury,
Bucks., HP20 2QZ.

Heinemann
Heinemann Computers in Education Ltd.,
22 Bedford Square,
London, W.C.1.

Henry
Brother Henry
Xaverian Brothers,
Bradley House,
Little Trodgers Lane,
Mayfield,
E. Sussex, TN20 6PW.

Hessel
Simon Hessel Software,
15 Lytham Court,
Cardwell Crescent,
Sunninghill, Berks.

Hill
Hill McGibbon Ltd.,
92 Fleet Street,
London, E.C.4.

Hutch
Hutchinson Software,
Hutchinson Education,
Hutchinson House,
17-21 Conway Street,
London W1P 6JD.

Hutton
Hutton Press Ltd.,
16 Howl Lane,
Hutton,
Driffield, YO25 9QA.

ILECC
Inner London Educational Computing Centre,
John Ruskin Street,
London, SE5 0PQ.

Incentive
Incentive Software Ltd.,
54 London Street,
Reading,
Berks. RG1 4SQ.

Island
Island Logic,
22 St. Peter's Square,
London, W6 9NW.

LCL
LCL,
26 Avondale Avenue,
Staines,
Middlesex, TW18 2PN.

LPA
Logic Programming Associates Ltd.,
10 Burntwood Close,
London, SW18 3JU.

LTS
Learning and Training Systems Ltd.,
Haydon Road,
Studley,
Warwickshire, B80 7AP.

Level 9
229 Hughenden Road,
High Wycombe,
Bucks., HP13 5PG.

Longman
Longman Micro Software,
33-35 Tanner Row,
York,
YO1 1JP.

MEP
Microelectronics Education
Programme,
Cheviot House,
The Polytechnic,
Coach Lane Campus,
Newcastle-upon-Tyne,
NE7 7XA.

MJL
M.J. Leonard,
24 Brockley Rise,
London, SE23 1PR.

MP
Micropax,
38 Garrick Green,
Old Catton,
Norwich, NR6 7AN.

MUSE
MUSE,
PO Box 43,
Hull,
HU1 2HD.

Macmillan
Macmillan Software,
Houndmills,
Basingstoke,
Hants.

Melbourne
Melbourne House,
131 Trafalgar Road,
London, SE10.

MicroPro
MicroPro,
1299 - 4th Street,
San Rafael,
CA 94901, U.S.A.
(Software available from retailers).

MicroTrust
MicroTrust Software,
National Extension College,
18 Brooklands Avenue,
Cambridge, CB2 2HN.

Micro
Micro Power Ltd.,
Northwood House,
North Street,
Leeds, LS7 2AA.

Mirror,
Mirrorsoft,
Holborn Circus,
London, EC1P 1DQ.

Molimerx
Molimerx Ltd.,
1 Buckhurst Road,
Town Hall Square,
Bexhill on Sea,
East Sussex.

Mosaic
John Wiley & Sons Ltd.,
Baffins Lane,
Chichester,
Sussex, PO19 1UD.

NCC
National Computing Centre,
Oxford Road,
Manchester, M1 7ED.

NCS
Network Computer Systems Ltd.,
39 Bampton Road,
Luton,
Beds., LU4 0DD.

NES
Nottingham Educational Supplies,
17 Ludlow Hill Road,
West Bridgford,
Nottingham, NG2 6HD.

Nelcal
Thomas Nelson & Sons Ltd.,
Nelson House,
Mayfield Road,
Walton-on-Thames,
Surrey, KT12 4BR.

Pitman
Pitmansoft,
128 Long Acre,
London, WC2E 9AN.

Primary
Primary Programs Ltd.,
Claypits,
Debden Road,
Saffron Walden,
Essex, CB11 3JS.

Psion,
2 Huntsworth Mews,
Gloucester Place,
London, NW1 6DD.

RML
Research Machines Ltd.,
Mill Street,
Oxford, OX2 0BW.

RoSPA
RoSPA,
The Priory,
Queensway,
Birmingham, B4 6BS.

S/C
Soft Centre,
Renryle Cottage,
Okehurst Lane,
Billinghurst,
West Sussex, RH14 9HR.

SPA
Software Production Associates,
PO Box 59,
Leamington Spa,
CV31 3QA.

SSS
Social Science Software,
FREEPOST,
Birkenhead,
Merseyside, L42 2AB.

Sharp
Sharpsoft,
90 Paul Street,
London, E.C.1.

Sherston
Sherston Software,
8 Court Street,
Sherston,
Malmesbury,
Wiltshire, SN16 0LL.

Sinclair
Sinclair Research Ltd.,
Stanhope Road,
Camberley,
Surrey, GU15 3PS.

Sloane
Sloane Software,
52 Belgrave Court,
Sloane Walk,
Shirley,
Croydon, CR0 7W.

Solent
Solent Software,
96 Coldharbour Road,
Redland,
Bristol, BS6 7SB.

Stack
Stack Computer Services Ltd.,
290-298 Derby Road,
Bootle,
Liverpool,
Merseyside, L20 8LN.

Sussex
Sussex Software,
Sussex Publications,
FREEPOST,
Devizes,
Wilts., SN10 1BR.

System
South Yorkshire Systems for Training
Education and Management Ltd.,
12 Collegiate Crescent,
Sheffield, S10 2BA.

Tasman
Tasman Software,
17 Hartley Crescent,
Leeds,
Yorkshire, LS6 2LL.

Tecmedia
Tecmedia Ltd.,
5 Granby Street,
Loughborough,
LE11 3DU.

Transdata
Transdata Ltd.,
Microelectronic Systems Division,
11 South Street,
Havant,
Hants., PO9 1BU.

Triptych
TBL Book Services,
17-23 Nelson Way,
Tuscam Trading Estate,
Camberley,
Surrey, GU15 3EU.

Wida
Wida Software,
2 Nicholas Gardens,
London, W5 5HY.

Appendix E

Useful addresses

CALICO Journal
233 SFLC,
Brigham Young University,
Provo,
UT 84602 U.S.A.

Centre for Information on Language Teaching and Research
(CILT)
20 Carlton House Terrace,
London, SW1Y 5AP.

Computers in Education as a Resource
(CEDAR)
Imperial College Computer Centre,
Exhibition Road, SW7 2BX.

Council for Educational Technology
3 Devonshire Street,
London, W1N 2BA.

ERIC Clearinghouse on Languages and Linguistics
(ERIC/CLL News Bulletin)
3520 Prospect Street NW,
Washington DC 20007 U.S.A.

Educational Computing
Priory Court,
30-32 Farringdon Lane,
London, EC1R 3AU.

Inner London Educational Computing Centre
(ILECC)
John Ruskin Street,
London, SE5 0PQ.

Investigations in Teaching with Microcomputers as an Aid
(ITMA)
College of St. Mark & St. John,
Derriford Road,
Plymouth, PL6 8BH.

Microcomputers in Computer Education
(MICE)
Polytechnic of the South Bank,
Borough Road,
London, SE1 0AA.

Microelectronics Education Programme
(MEP)
Cheviot House,
Coach Lane Campus,
Newcastle-upon-Tyne,
NE7 7XA.

Microcomputer Users in Schools
(MUSE)
PO Box 53,
Hull.

Microcomputers and Primary Education
(MAPE)
76 Sudbrooke Holme Drive,
Sudbrooke,
Lincoln, LN2 2SF.

Teachers of English to Speakers of Other Languages
(TESOL)
202 DC Transit Bldg,
Georgetown University,
Washington DC 20057 U.S.A.

BEST COPY AVAILABLE

Further copies available from:
Adult Literacy & Basic Skills Unit,
Kingsbourne House, 229/231 High Holborn,
London WC1V 7DA.
Tel: 01-405 4017

Price
£2.50

ALBSU
Adult Literacy & Basic Skills Unit

ilea